TECHNICAL REPORT NO. 642

ANALYSIS AND APPLICATION OF THE PADE' APPROXIMATION FOR THE INTEGRATION OF CHEMICAL KINETIC EQUATIONS

By D. Magnus
H. Schecter

March 30, 1967

Sponsored by

Advanced Research Projects Agency
Missile Phenomenology Branch
ARPA Order No. 396

GENERAL APPLIED SCIENCE LABORATORIES, INC.
A SUBSIDIARY OF THE MARQUARDT CORPORATION
MERRICK AND STEWART AVENUES
WESTBURY, L. I., NEW YORK

Project 5810

Total No. of Pages - iv and 94

Copy No. ([redacted]) of 125

TECHNICAL REPORT NO. 642

ANALYSIS AND APPLICATION OF THE PADE' APPROXIMATION FOR THE INTEGRATION OF CHEMICAL KINETIC EQUATIONS*

By D. Magnus
H. Schecter

Prepared for

Advanced Research Projects Agency
Washington, D. C.

Contract Number DA-49-083 OSA-3135
ARPA ORDER No. 396

"Missile Phenomenology Studies"

Prepared by

General Applied Science Laboratories, Inc.
A Subsidiary of The Marquardt Corporation
Merrick and Stewart Avenues
Westbury, L.I., New York

*This research was supported by the Advanced Research Projects Agency, Missile Phenomenology Branch, and was monitored by Mr. C. E. McLain, under Contract No. DA-49-083 OSA-3135.

March 30, 1967

Approved by: Lewis Feldman
LEWIS FELDMAN, MANAGER
APPLIED RESEARCH DEPARTMENT

ABSTRACT

A stable numerical technique is given for the integration of the fluid flow equations involving finite rate chemical kinetics. The procedure permits a significantly larger integration step than standard procedures, such as Runge-Kutta. Consequently, the amount of digital computer time to numerically integrate the equations over a specified domain is greatly reduced. To simplify the implementation of the numerical procedure for generic chemical systems, a programming system has been developed that automatically generates the necessary subroutines from a few inputs. The extension and application of the numerical method to partial differential equations is also discussed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TECHNICAL REPORT NO. 642

ANALYSIS AND APPLICATION OF THE PADE'APPROXIMATION FOR THE INTEGRATION OF CHEMICAL KINETIC EQUATIONS

By D. Magnus
H. Schechter

1. INTRODUCTION

For the analysis and study of flow problems involving finite rate chemical reactions, the equations of mass conservation, momentum, state, energy, and species conservation must be solved. The latter equation although appearing simple in form introduces severe difficulties in the numerical solution of such problems. In the case of one-dimensional flow problems, the species conservation relationships for n species are represented by n-coupled non-linear ordinary differential equations (ODE). Numerous standard numerical methods can be used to integrate these equations; these procedures will be severely limited by the size of the integration step. For two-dimensional problems, a coupled system of partial differential equations represents the species conservation relationship, and the numerical integration of these equations is likewise very difficult using standard explicit or implicit methods. Usually, the implicit method is considered unconditionally stable, and therefore an integration step of any size can bo used within the limitation imposed by truncation error. Unfortunately, this property of the implicit method is dependent upon the form of the differential equations. Because of the coupling of the n species in each conservation equation, the amplification matrix is not necessarily convergent, and the method can become unstable unless very small steps are used (see Section 5).

Methods of overcoming the limitation on stepsize have been attempted by numerous investigators. In Ref. 7, the authors develop a very useful method of integrating a first order differential equation of chemical kinetics. The procedure is limited to uncoupled equations, and quite naturally other authors have attempted to extend this procedure to coupled systems through the use of suitable transformationn. However, the transformation required considerable computation which had to be repeated at each step. Other investigators (Ref. 8) modified the Runge-Kutta procedure and improved its stability behavior under certain circumstances. In Ref. 9, a review of some of these approaches is presented.

For this study, the method of rational approximations, which was introduced by GASL (Ref. 3), has been selected as most suitable for integrating the species conservation equations. The details of the method are outlined in Section 2, and it is demonstrated that the method is unconditionally stable. The development is related to ODE, but in Section 5 a method of applying the technique to partial differential equations is discussed. The method has been successfully used in numerous types of problems involving as many as 31 species and 78 reactions.

Besides being stable, the numerical method must be convenient to use, if it is to be accepted by the engineer and scientist. Since the method of rational approximations involves rather lengthy algebra, an algorithm for automatically implementing the method for a generic chemical system has been developed as part of this investigation. Further, the algorithm has been programmed and made operational in conjunction with the GASL Generalized Chemistry Program. With this sytem approach, the user merely specifies the desired species and chemical reactions, and the necessary subprograms are automatically generated by the machine. A completely debugged subroutine in FORTRAN IV is supplied as output from this system. The system is very flexible and if either a species or a reaction is not resident in the chemistry library, it can be readily added. The capacity of the system is now 100 species and 200 reactions. In Section 3 of this report, the details of this system are discussed.

The advantayes of using the method of rational approximations is demonstrated in Section 4; the results for two different dissociating air systems (7 and 11 species) are presented.These results are compared with the results from the Runge-Kutta integration procedure, and the agreement is good. For the problems studied, the method of rational approximation required approximately 1/20 to 1/25 of the machine time used by the Runge-Kutta procedure. With this very significant reduction in machine time, the engineer can economically investigate large chemical systems without resorting to approximations or simplifications regarding the number of reactions or species.

2. STABLE INTEGRATION PROCEDURES

A system of n first order ordinary differential equations can be expressed in general form as

$$y'(t) = f(t,y) \quad ; \quad y(t_o) = y_o \qquad (2.1)$$

where y is a vector valued function of t, and f is a vector valued function of t and y. Given the initial vector y_o, the system of ordinary differential equations is to be numerically integrated over the time domain (t_o,t_f).

In this section, the numerical procedures that can be used to integrate (2.1) are considered. In particular the stable integration procedures are discussed. ıor the purpose of this report, a numerical method of solution is termed stable if an error, once introduced, decreases from step to step when the eigenvalues of the matrix $[\partial f/\partial y]$ have negative real parts (see Ref. 3).

2.1 Discussion of Error Propagation

Many methods have been devised for obtaining a numerical solution of (2.1). However, most of these methods will not give a valid solution if the stepsize h becomes too large. Although the truncation error increases in a smooth manner as h increases, the numerical solution becomes unstable when h exceeds a certain critical value. Even if the truncation or round-off errors are small, the propagated error becomes unbounded because the amplification term in the error expression has a spectral radius greater than one.

The unstable behavior can be illustrated and understood by considering the error behavior of the Euler method. Let

$$\bar{y}_{k+1} = \bar{y}_k + h\bar{f}_k \qquad (2.2)$$

where $f_k = f(t_k, \bar{y}_k)$, h is the stepsize between t_{k+1} and t_k, and $t_k = kh$. The stepsize, h, can be assumed a constant for this discussion. The true solution can be expressed as

$$y_{k+1} = y_k + hf_k + \tau_{k+1} \qquad (2.3)$$

where τ_{k+1} is the truncation error. By subtracting (2.2) from (2.3), applying the mean value theorem, and letting $e_k = y_k - \bar{y}_k$, the error expression becomes

$$e_{k+1} = \left(I + h\frac{\partial f}{\partial y}\right) e_k + \tau_{k+1} \qquad (2.4)$$

where $\frac{\partial f}{\partial y}$ is the matrix $\left[\frac{\partial f_i}{\partial y_j}\right]$. The operator on the error term e_k is the amplification matrix which determines the stability behavior of the system of equations. Dropping the truncation error and assuming that $\frac{\partial f}{\partial y}$ is a constant, an initial error vector e_o will propagate as

$$e_{k+1} = \left(I + h\,\frac{\partial f}{\partial y}\right)^{k+1} e_o \qquad (2.5)$$

If the eigenvalues of $\frac{\partial f}{\partial y}$ have positive real parts, the solution would be increasing and the numerical error could likewise be expected to increase. If, however, the eigenvalues all have negative real parts and an error increases from step to step, the numerical solution will then be partially unstable.

If the error generated by Euler's method is not to increase, the eigenvalues of the amplification matrix $I + h\,\frac{\partial f}{\partial y}$ must be less than one in absolute value. Otherwise, as the matrix is raised to higher powers, the norm of the error vector will continually grow. Let the ℓ^{th} complex eigenvalue of $\frac{\partial f}{\partial y}$ be

$$\lambda_\ell = -\,\alpha_\ell + i\beta_\ell$$

where $\alpha_\ell > 0$. Then by imposing the stability requirement, a sufficient condition for stability is

$$h < 2/\hat{\alpha}$$

where $\hat{\alpha} = \max_{\ell} \left|\mathcal{R}\lambda_\ell\right|$.

If $\hat{\alpha}$ is large, as in the chemical kinetic case, this requirement may result in a small h and a large amount of computing time if the domain of integration is large. The above behavior for the Euler method is characteristic of most standard single and multi-step methods. Extending the analysis to the more complicated methods does not present any difficulty other than cumbersome algebra and has been reported in Ref. 4. For the purpose of later discussion, the stability criterion for the fourth order Rung-Kutta method is

$$h < 2.8 \Big/ |\lambda|$$

where λ is an eigenvalue of maximum modulus.

The equations describing the chemical reactions for dissociating air can be used to illustrate the restrictions on stepsize. For a streamtube type of calculation for a reentry body (see Section 4), the dominant eigenvalue is about 3×10^{4}, and for the Euler method to remain stable, $h < .6 \times 10^{-4}$ (ft). To integrate the equations over a distance of 0.5 ft would require about 8000 steps. These small steps are required even though the solutions change very little from one step to the next.

If the stepsize limitation imposed by the stability could be eliminated, then substantially fewer steps (and less machine time) would be required. As shown in the next subsection, the limitation can be removed by using numerical methods which are unconditionally stable.

2.2 The Pade' Integration Method

A class of stable integration formulas can be derived by using known properties of linear ODE. These formulas, while directly applicable to a system of linear ODE, may be applied to systems of non-linear differential equations by linearizing at each step. Even considering the additional computer time necessary to linearize the equations, the total time needed to solve a complete problem is much less than required by any standard method.

Consider a system of n linear ODE with constant coefficients

$$\dot{y}(t) = A\, y(t) + b \quad y(t_o) = y_o \tag{2.6}$$

The formal solution of this equation at t=h is

$$y(h) = \exp(hA)\,[y_o + A^{-1} b] - A^{-1} b \tag{2.7}$$

where

$$\exp(hA) = I + hA + \frac{1}{2}(hA)^2 + ---$$

In Eq. (2.7), the exponential term can be approximated in numerous ways, and the form of the approximation will determine whether the procedure is stable. For our purposes, a convenient method of approximating any function is the rational approximation, which can be derived using the Pade' transformation of the power series. If P and Q are polynomials of degree p and q, respectively, the exponential is represented as

$$e^{hA} = Q^{-1} P + E(hA)$$

The rational approximation agrees with the power series of exp (hA) for at least p + q + 1 terms and the residual error is denoted by E(hA). The polynomials P and Q for the exponential function (Ref. 5) are given by

$$P = \sum_{k=0}^{p} \frac{(p + q - k)!\ p!}{(p + q)!\ k!\ (p - k)!} (hA)^k$$

$$Q = \sum_{k=0}^{q} \frac{(p + q - k)!\ q!}{(p + q)!\ k!\ (q - k)!} (-hA)^k$$

Substituting the rational approximation into formula (2.7) yields the desired integration formula:

$$y(h) = Q^{-1} P (y_o + A^{-1} b) - A^{-1} b$$

Or rearranging the terms,

$$y(h) = Q^{-1}[Py_o + (P - Q) A^{-1} b] \qquad (2.8)$$

Equation (2.8) includes numerous types of single step integration methods and not all of these methods are stable. The Euler method, which has been shown to be partially stable in the previous section, is a special case of (2.8) with $Q = I$ and $P = I + hA$ ($q = 0$ and $p = 1$). In this report, only the stable procedures are of interest and for these methods $q \geq p$. Of particular interest, are those methods where the diagonal ($p = q$) of the Pade' table is used. Such methods shall be called Pade' integration procedures or rational approximation methods in this report. For the cases $p = q = 1$ and $p = q = 2$, the following expressions are obtained from (2.8):

$$y(h) = (I - \frac{1}{2} hA)^{-1} [(I + \frac{1}{2} hA)y_o + hb] \qquad (2.9)$$

$$y(h) = \left(I - \frac{1}{2} hA + \frac{1}{12} (hA)^2\right)^{-1} \left[\left(I + \frac{1}{2} hA + \frac{1}{12} (hA)^2\right)y_o + hb\right]$$

Since these formulas must agree with the power series for at least $p + q + 1$ terms, their truncation errors are immediately seen to be $O(h^3)$ and $O(h^5)$, respectively. Similarly, integration formulas can be derived for any order of truncation error.

The stability of the Pade' integration method can be investigated by considering the following form of (2.8)

$$\bar{y}(h) = Q^{-1} P y_o + Q^{-1} Rb$$

where R represents some polynomial in (hA). Applying the method in a step by step manner yields

$$\bar{y}_{k+1} = Q^{-1} P \bar{y}_k + Q^{-1} Rb \qquad (2.10)$$

where $y_k = y(kh)$. The true solution y can also be expressed as

$$y_{k+1} = Q^{-1} P y_k + Q^{-1} Rb + \tau_{k+1} \qquad (2.11)$$

where τ_{k+1} is the truncation error. Denote the error at each step by

$$e_k = y_k - \bar{y}_k$$

and subtract (2.10) from (2.11) to obtain

$$e_{k+1} = (Q^{-1} P)e_k + \tau_{k+1} \qquad (2.12)$$

In order to see how an error from any source propagates, we may neglect the truncation error term and assume that a single error e_o is cormitted at one step, say the first step. Equation (2.12) then becomes

$$e_k = Q^{-1} P e_{k-1} = (Q^{-1} P)^2 e_{k-2} = \cdots = (Q^{-1} P)^k e_o \quad (2.13)$$

After k steps an initial error e_o has increased or decreased, depending upon the amplification matrix $(Q^{-1} P)$.

From the definition of stability, when the eigenvalues of the matrix A have negative real parts, e_k must decrease as k increases. Since e_o is a constant, e_k can decrease only if the matrix $(Q^{-1} P)^k$ is convergent; that is, the eigenvalues of $(Q^{-1} P)$ must be less than one in absolute value. For every eigenvalue λ_i of the matrix A in (2.6), there is a corresponding eigenvalue for $Q^{-1} P$. For convenience, the stability criteria is stated in terms of λ by

$$\| P(\bar{\lambda})/Q(\bar{\lambda}) \| < 1$$

Thus the stability requirement becomes

$$|P(\bar{\lambda})| < |Q(\bar{\lambda})| \quad (2.14)$$

where the complex eigenvalue $\bar{\lambda} = h\lambda = -\alpha + i\beta (\alpha > 0)$. For $p = q$ it can be shown that the stability requirement is satisfied.

The Pade' integration method is directly applicable to linear systems of ordinary differential equations. Some additional steps are required to apply the method to chemical kinetic equations, which are represented by (2.1). The non-linear system of equations can be reduced to the form (2.6) by expanding the function f in a Taylor's Series and neglecting high order terms:

$$\dot{y} = f(t_o,y_o) + \frac{\partial f}{\partial y}(t_o,y_o)\ \Delta y + O(\Delta y^2) \qquad (2.15)$$

where $\Delta y = y - y_o$ and $\frac{\partial f}{\partial y}(t_o,y_o)$ denotes the matrix A in (2.6) with elements

$$a_{ij} = \frac{\partial f_i}{\partial y_j}$$

The indices i and j denote the i^{th} and j^{th} elements in the f and y vectors, respectively. Hence grouping terms as in (2.6)

$$y = \frac{\partial f}{\partial y}(t_o,y_o)y + \left[f(t_o,y_o) - \frac{\partial f}{\partial y}(t_o,y_o)y_o\right] + O(\Delta y^2)$$

$$= Ay + b + O(\Delta y^2) \qquad (2.16)$$

Since the equation must be linearized at each step of the integration, the residual error, which is denoted by $\zeta_{k+1} = O(\Delta y^2)$ for the k^{th} step, must be introduced in Eq. (2.12),

$$e_{k+1} = (Q^{-1}\ P)e_k + Q^{-1}\ R\ \ell_{k+1} + \tau_{k+1} \qquad (2.17)$$

Again, assuming the matrix A and the vector b are nearly constant over (k+1) steps then

$$e_{k+1} = (Q^{-1}\ P)^{k+1} \left[e_o + \sum_{\nu=1}^{k+1} (Q^{-1}\ P)^{-\nu}\ (Q^{-1}\ R\ \ell_\nu + \tau_\nu) \right] \qquad (2.18)$$

Equation (2.18) which is similar to (2.13) includes the influence of the truncation error and the residual error of linearization. Again, the amplification matrix $Q^{-1}P$ must be a convergent matrix for a stable system. The rate of decay of $(Q^{-1}\ P)^k$ and the magnitude of the error $Q^{-1}\ R\ \ell_k + \tau_k$ introduced at each step will determine whether e_{k+1} is decreasing with k. Since the linearization and the truncation errors appear together, the integration method for a particular type of problem should be selected to achieve a truncation error approximately the same size as the linearization error. Hence, if a large residual error is present, a low order integration formula should be used.

3. IMPLEMENTATION OF THE INTEGRATION PROCEDURES

As shown in Section 2, the Pade' approximation technique for solving systems of ordinary differential equations is directly applicable to linear equations. For non-linear systems of equations, the functions $f_i(y_j, t)$; i, j = 1, ... n where n is the number of species in the chemical system must be expanded in a Taylor's Series. The terms of second order and higher are neglected in the series for each function. At each step of the integration, this linearized form of the functions must be evaluated. Hence, when the Pade' method of integration is implemented on a digital computer it is convenient to develop two separate routines. The first routine calculates the linearized form of the functions f_i and the second routine integrates the linear form over a single step in the independent variable. Using the terminology of Section 2 [Eq. (2.16)], the former routine calculates the matrix, A, and forcing vector, b, whereas the latter routine solves a system of linear first order ordinary differential equations.

The system of linear ordinary differential equations is defined by the matrix, A. The routine which solves this system is related to the chemistry problem only through the size of the A matrix. Hence, this integration routine need only be written once provided the size of the A-matrix can be varied as the number of species is changed in the chemical system. Such a general purpose integration routine has been written and combined with the one-dimensional flow equations to form a general program for the analysis of streamtube problems involving chemical kinetics. This program and its use is described in Subsection 3.3.

The subroutine for forming the matrix, A, and forcing vector, b, is directly dependent upon the types of reactions and species, and consequently, this routine must be recoded for each new chemistry system. For a large number of reactions and species, the analysis and coding are very tedious and error prone. To overcome this difficulty an algorithm has been developed for forming A and b given a generic chemistry system. Furthermore, the algorithm was implemented in conjunction with the Generalized Chemistry Program (GCP) developed at GASL (see Ref. 1). Given a small number of inputs, the GCP generates in FORTRAN language the subroutines which can be used with the integration program.

In Section 3.1, the algorithm for forming the A-matrix and b-vector is described along with the programming procedures. A general description of the GCP is given in Section 3.2, and the required input cards for generating the subroutine are described. The GCP has been successfully used on a variety of chemistry problems involving as many as 31 species and 78 reactions.

3.1 Algorithm for the Linearized Form

Problems in chemical kinetics give rise to differential equations of the form:

$$\dot{y}_i = \sum_{j=1}^{m} (\nu''_{ij} - \nu'_{ij})\, k_j(T)\,\rho^{-1} \prod_{\ell=1}^{n} (\rho y_\ell)^{\nu'_{\ell j}} \qquad i=1,2,\ldots n \qquad (3.1)$$

where n is the number of reacting species, m is the total number of reactions (the sum of forward and reverse reactions), y_i is the number of moles per unit mass of the mixture, and the dot denotes differentiation with respect to time. T and ρ are the temperature and density, respectively. The stoichiometric coefficients ν'_{ij} and ν''_{ij} are defined by the following expression for the jth chemical reaction equation:

$$\sum_{i=1}^{n} \nu'_{ij} R_i \rightarrow \sum_{i=1}^{n} \nu''_{ij} P_i \qquad (3.2)$$

where R_i and P_i are the reactants and products, respectively. The reaction rate for the jth reaction is k_j which is taken as a function of temperature only and has the form

$$k_j(T) = a_j\, T^{b_j} \exp\,(-e_j/T)$$

where a_j, b_j, and e_j are constants.

The right side of Eq. (3.1) represents the function f_i which must be expanded in a Taylor's Series as shown in Eq. (2.15). In terms of the functions f_i the elements of the matrix, A, and vector, b, are

$$a_{is} = \frac{\partial f_i}{\partial y_s} \tag{3.3}$$

$$b_i = f_i(t_o, y_o) - \sum_{k=1}^{n} a_{ik}(y_o)_k \tag{3.4}$$

where index i denotes the species production equation, index s denotes the species, and $(y_o)_k$ is an element of the vector of species at time t_o. The algorithm which generates the program for (3.3) and (3.4) requires numerous constants (e.g. a_j, b_j, e_j, ν''_{ij}, etc). All of the required constants are on the library tape of the GCP which was developed by GASL. The algorithm for handling the constants, defining the arrays, and operating upon these arrays is described below.

When a particular set of species is selected, the species are consecutively numbered according to their order of input. Each speciesis then represented by the corresponding integer throughout the logic of the generator program. These integers are used in a 12 x m/2 array, which represents the reactions (3.2). The first six columns of the array are used for the left-hand side of the reactions and the last six columns represent the right-hand side. The integers corresponding to species with non-zero stoicniometric coefficients in the reaction equation are placed in these columns. If a stoichiometric coefficient is greater than one, the specie number is simply repeated the proper number of times. For example, the reaction

$$R_5 + 2R_{12} + R_{18} \rightarrow P_7 + P_{22}$$

would occupy one row of the reaction array and would appear in storage as (each box is a storage cell):

5	12	12	18	0	0	7	22	0	0	0	0

Note the reactant R_{12} occupies two memory cells because its stoichiometric coefficient is two in the reaction equation. Since all reactions are considered reversible (if not the reverse reaction rate is merely set equal to zero), the reverse reaction is obtained by merely scanning the row from right to left.

As indicated above, the current form of the logic of the generator program is restricted to reaction equations involving 6 or less reactants. Four of these reactants may be different species. The same restrictions are placed upon the products. Within these restrictions our survey indicates that all reactions of current interest in the area of reentry physics can be included. If in the future more sophisticated chemical reactions are required (e.g. 5 body reactions or reactions involving large stoichiometric coefficients), the program logic is readily extended.

The reaction array is subdivided into three distinct groups. Each reaction which is to be stored in memory is first classified, and then assigned to the corresponding group. The first group consists of all of the noncatalytic reactions. The second group is for nondesignated catalytic reactions; that is, the reaction rate is independent of the species which are used as catalysts. In the third group, the designated catalytic reactions appear. For these reactions, the reaction rate is dependent upon the species designated as a catalyst. There may be $p (p \leq 10)$ different reaction rates for each designated catalytic reaction. Consequently, the set of species must be partitioned into p or less subsets for each reaction in the third group. An auxiliary array is used to define these subsets and their corresponding reaction rates.

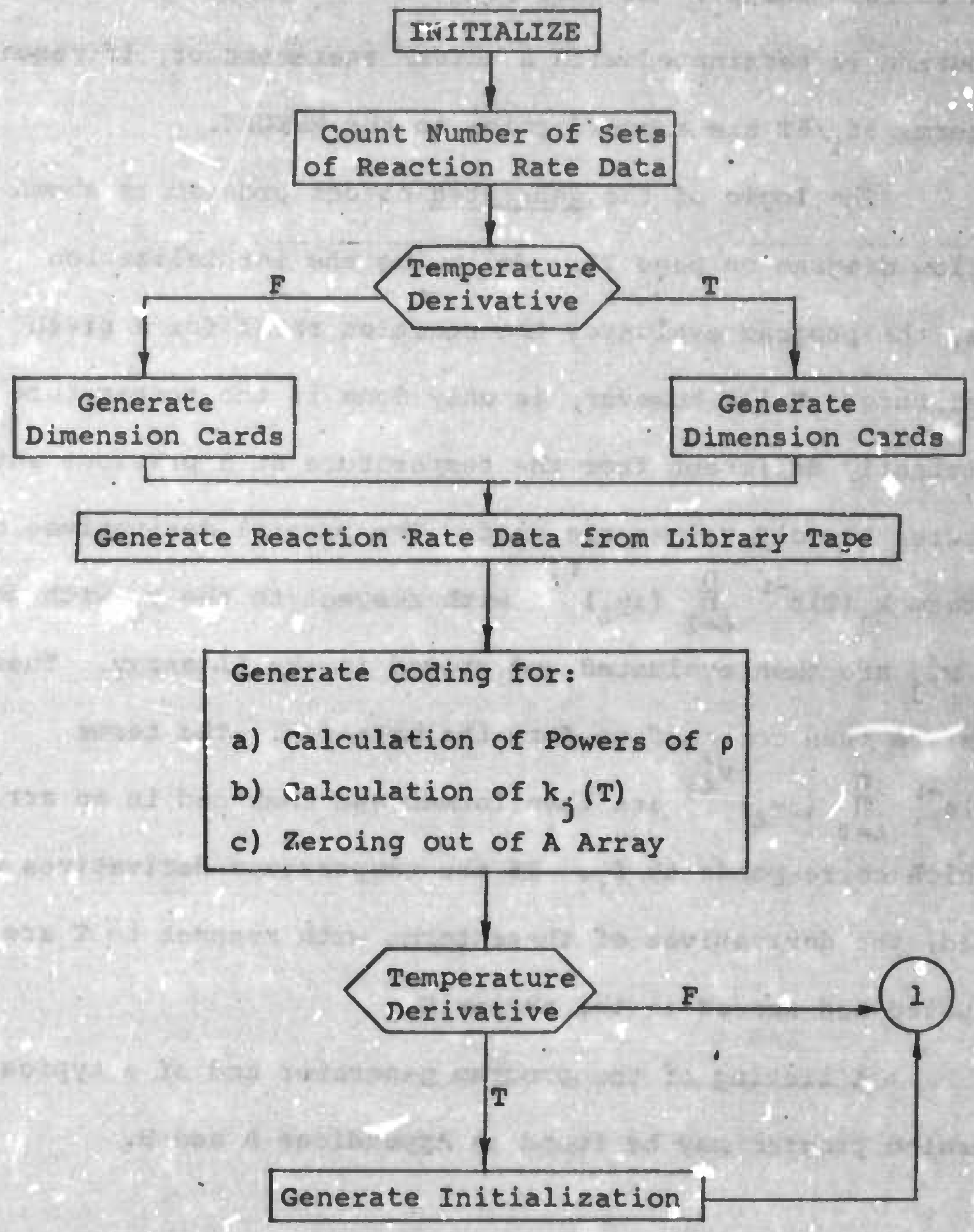
All the information that is necessary to form the functions f_i and then differentiate them with respect to each of the species as implied by Eqs. (3.3) and (3.4), is contained in the arrays described above. These arrays are initially formed by the program generator, and then processed as shown in the flow diagram on page 24 to encode the desired program. There are two variations of the object program depending upon whether or not the derivative of f_i with respect to temperature, T, is needed (see Subsection 3.3). If they are needed,an array for these values is added to the dimension cards. Coding is then generated to define the reaction rate constants, calculate powers of ρ needed in subsequent calculations, and calculate the reaction rates for the given temperatures. Coding is also generated to initialize the A array to zero,and if the temperature derivative is needed an appropriate indicator is generated.

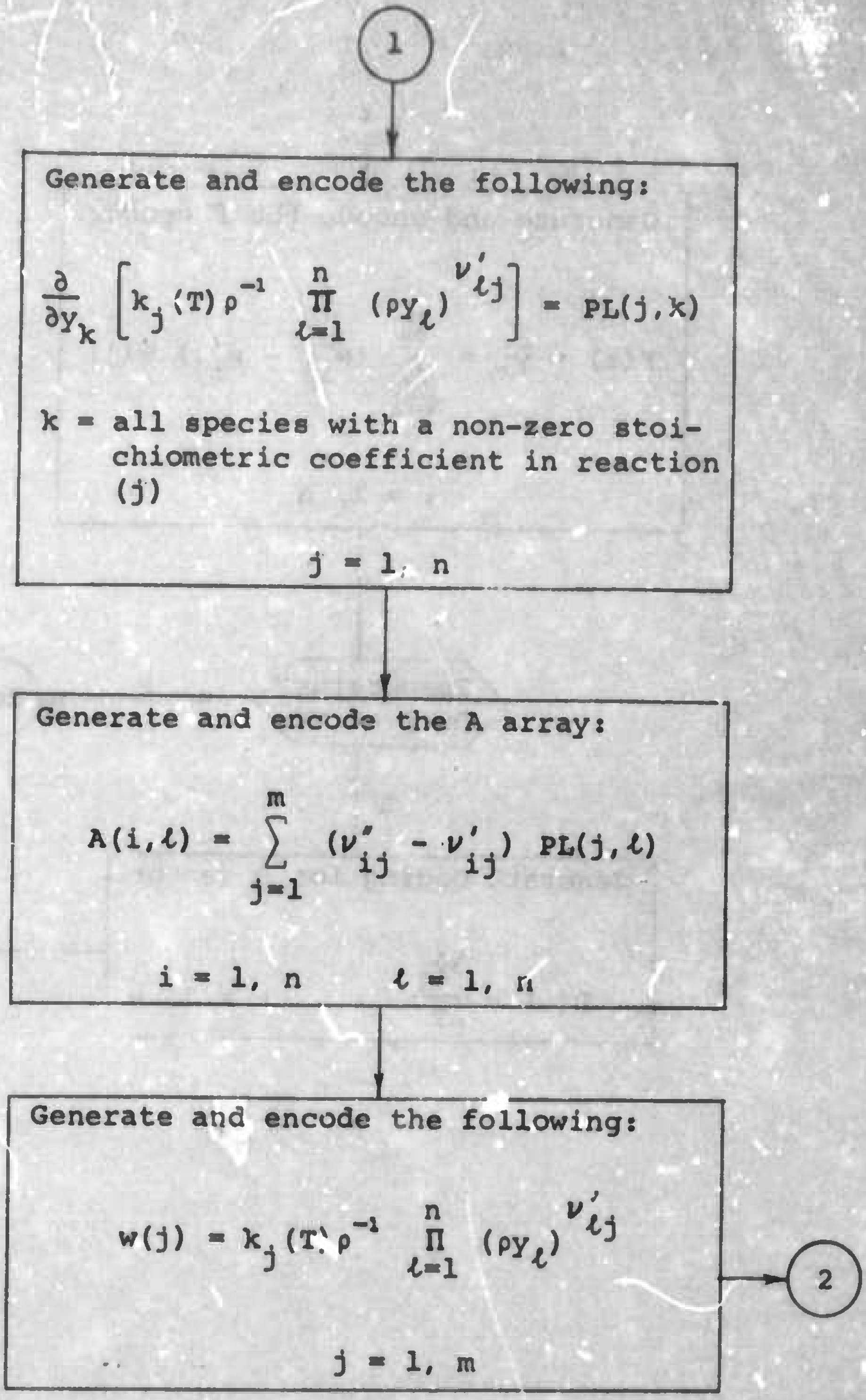
The next phase in the flow diagram is the generation of the code for the partial derivatives of each reaction with respect to each species that has a nonzero stoichiometric coefficient. This coding is used to define the two-dimensional array PL. Coding is then produced to evaluate the terms of the A matrix from combinations of the terms in the PL array. The terms $k_j(T)\rho^{-1}\prod_{\ell=1}^{n}(\rho y_\ell)^{\nu'_{\ell j}}$ are then encoded, and the coding produced to form the f_i. At this point in the program, either the subroutine is terminated with a RETURN statement or, if required, the terms $\partial f_i/\partial T$ are encoded prior to the RETURN.

The logic of the generated object program is shown in the flow diagram on page 27. Following the initialization phase, the program evaluates the reaction rates for a given temperature. This, however, is only done if the temperature is sufficiently different from the temperature at a previous entry. Otherwise the old values are used. The partial derivatives of the term $k_j(T)\rho^{-1}\prod_{\ell=1}^{n}(\rho y_\ell)^{\nu'_{\ell j}}$ with respect to the y_ℓ with non-zero $\nu'_{\ell j}$ are then evaluated and stored in the PL array. These terms are then combined to form the matrix A. The terms $k_j(T)\rho^{-1}\prod_{\ell=1}^{n}(\rho y_\ell)^{\nu'_{\ell j}}$ are then formed and combined in an array F, which corresponds to f_i. If the temperature derivatives are needed, the derivatives of these terms with respect to T are evaluated and stored in the vector D.

A listing of the program generator and of a typical generated program may be found in Appendices A and B.

FLOW DIAGRAM FOR THE GENERATOR PROGRAM


INITIALIZE
Count Number of Sets of Reaction Rate Data
Temperature Derivative
F
T
Generate Dimension Cards
Generate Dimension Cards
Generate Reaction Rate Data from Library Tape
Generate Coding for:
a) Calculation of Powers of ρ
b) Calculation of $k_j(T)$
c) Zeroing out of A Array
Temperature Derivative
F
1
T
Generate Initialization



Generate and encode the following:

$$\frac{\partial}{\partial y_k}\left[k_j(T)\rho^{-1}\prod_{\ell=1}^{n}(\rho y_\ell)^{\nu'_{\ell j}}\right] = PL(j,k)$$

k = all species with a non-zero stoichiometric coefficient in reaction (j)

j = 1, n

Generate and encode the A array:

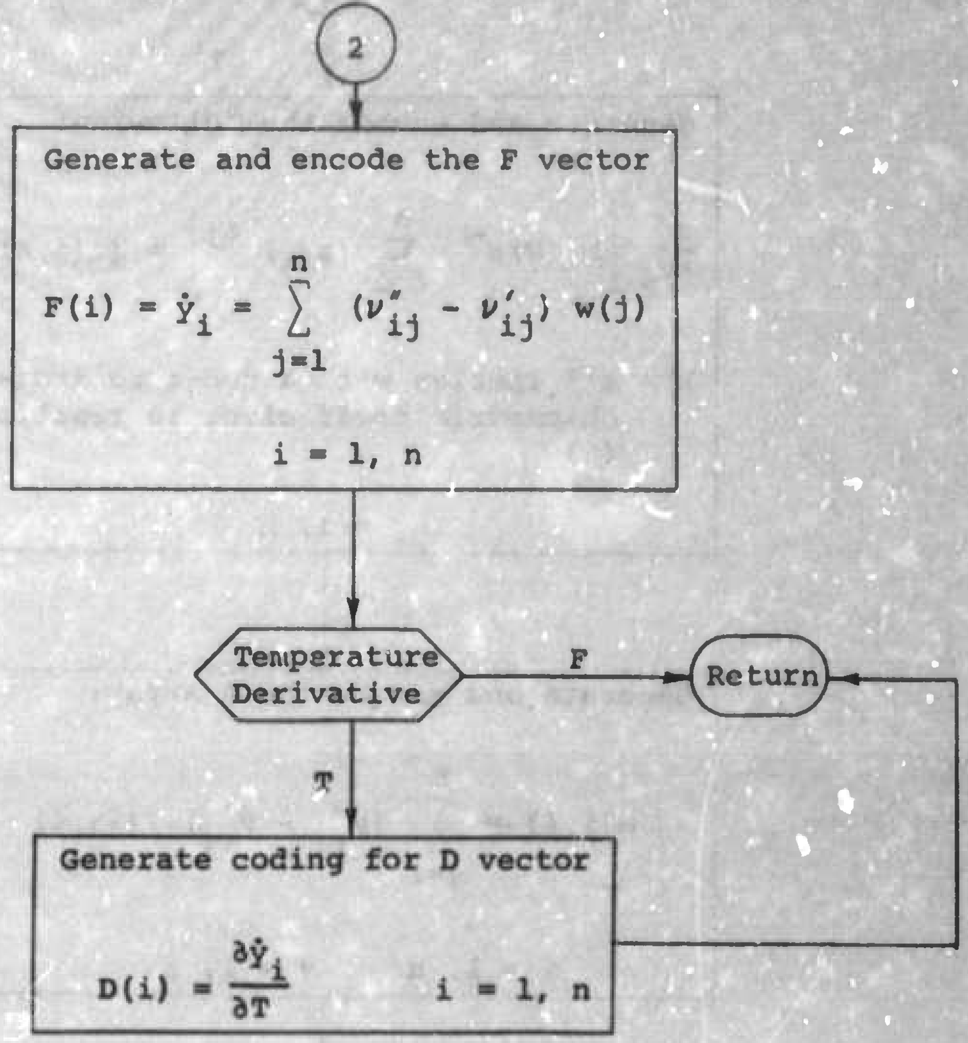
$$A(i,\ell) = \sum_{j=1}^{m}(\nu''_{ij} - \nu'_{ij})\ PL(j,\ell)$$

i = 1, n ℓ = 1, n

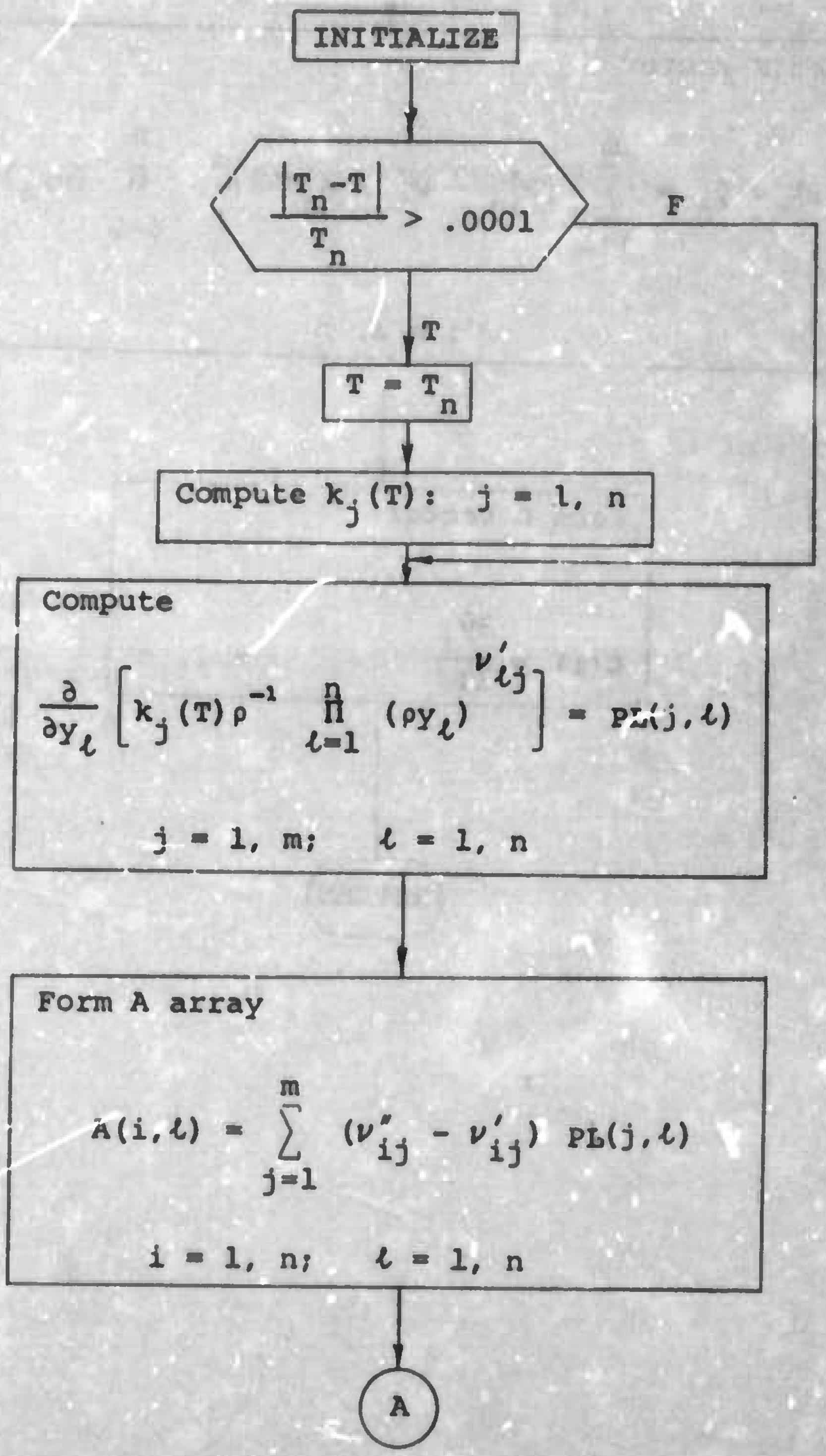
Generate and encode the following:

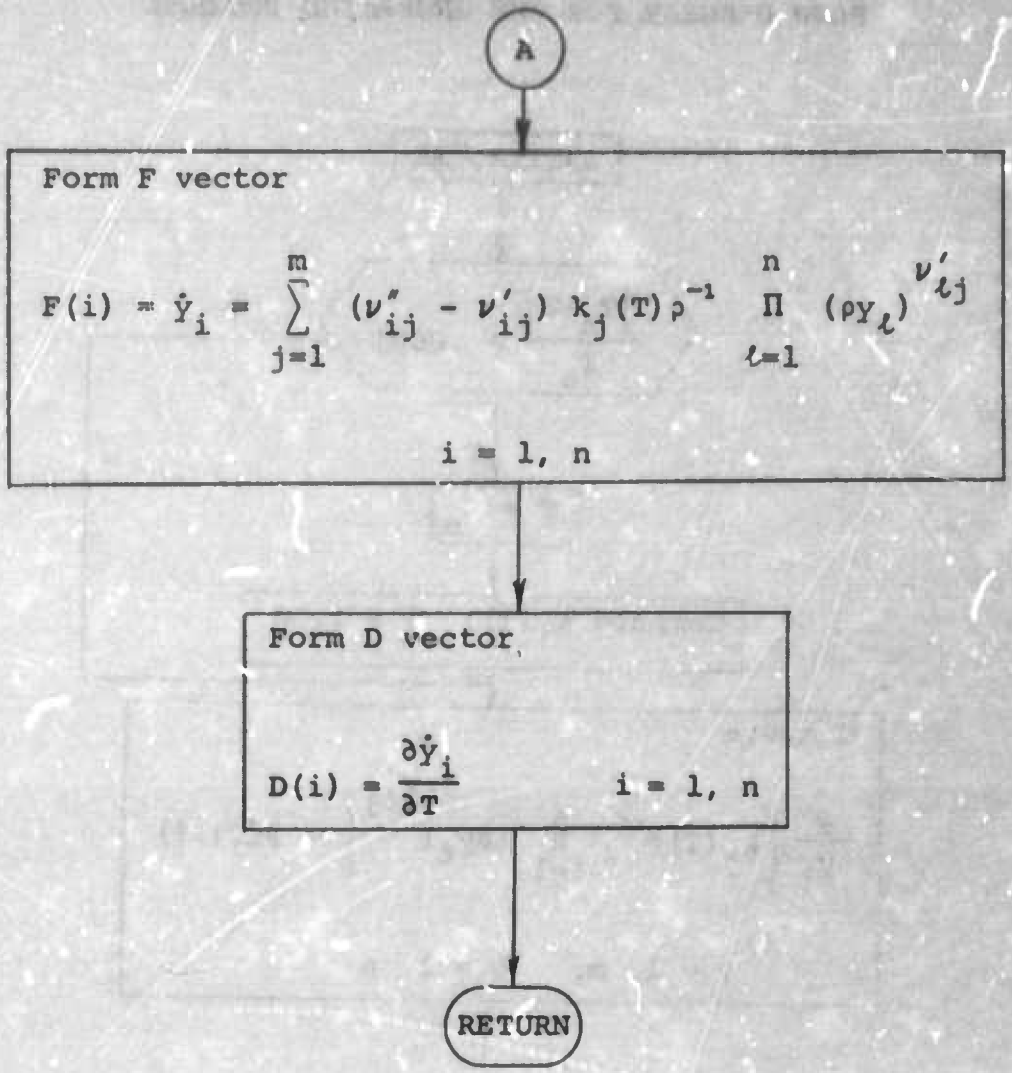
$$w(j) = k_j(T)\rho^{-1}\prod_{\ell=1}^{n}(\rho y_\ell)^{\nu'_{\ell j}}$$

j = 1, m


2
Generate and encode the F vector
$F(i) = \dot{Y}_i = \sum_{j=1}^{n} (\nu''_{ij} - \nu'_{ij}) w(j)$
i = 1, n
Temperature Derivative
F
Return
T
Generate coding for D vector
$D(i) = \frac{\partial \dot{Y}_i}{\partial T}$
i = 1, n

FLOW DIAGRAM FOR THE GENERATED PROGRAM


INITIALIZE
$\frac{|T_n - T|}{T_n} > .0001$
F
T
$T = T_n$
Compute $k_j(T)$: $j = 1, n$
Compute
$\frac{\partial}{\partial y_\ell}\left[k_j(T)\rho^{-1} \prod_{\ell=1}^{n} (\rho y_\ell)^{\nu'_{\ell j}}\right] = PL(j,\ell)$
$j = 1, m; \quad \ell = 1, n$
Form A array
$A(i,\ell) = \sum_{j=1}^{m} (\nu''_{ij} - \nu'_{ij})\ PL(j,\ell)$
$i = 1, n; \quad \ell = 1, n$
A



Form F vector

$$F(i) = \dot{Y}_i = \sum_{j=1}^{m} (\nu''_{ij} - \nu'_{ij})\, k_j(T)\, \rho^{-1} \prod_{\ell=1}^{n} (\rho Y_\ell)^{\nu'_{\ell j}}$$

$i = 1, n$

Form D vector

$$D(i) = \frac{\partial \dot{Y}_i}{\partial T} \qquad i = 1, n$$

3.2 Use of the General Chemistry Program (GCP)

In this subsection, the procedure for generating a program for the linearized model of the production equations for finite-rate chemical systems is described. Emphasis is placed on the input format, operating instructions and documentation of the GCP. A description of the GCP together with sample inputs and outputs is also included.

3.2.1 General Description of the GCP

The GCP is a modular series of programs which interact to generate other programs. As shown in Fig. 1, after accepting the input quantities the system can take three general types of action. The supervisor program will select a subprogram to modify or update the library tape, to generate a report or to generate one or more FORTRAN IV subroutine source decks. The former two options are actually support functions for the latter option; that is, the generation of source decks is the primary objective of the GCP. Reporting upon the contents of the source decks and maintaining a library of chemical kinetic data are necessary activities to achieve this objective.

The library tape contains data for various species and reactions involving these species. For each species, the library tape contains its chemical symbol, molecular weight, cubic polynomial fits of enthalpy and entropy for various temperature ranges, and coefficients for mixture viscosity, binary diffusion, specific heat and thermal conductivity models. For each chemical reaction, the library stores the complete reaction equation, the forward and backward reaction rates, and an indicator that denotes if the reaction is catalytic. If the reaction is catalytic, it is further classified into one of two groups. In one of the groups all species act as catalysts with a common reaction rate; in the other group the species (specified by the user) are separated into ten or less designated sets and each set has a different reaction rate.

The library can be modified by providing the proper control cards to the supervisor program as indicated in Fig. 1. Then the program control is given to the library update program which reads the new information or modifications of data into the primary memory of the computer. A new master library tape is then created by merging the original tape and the modifications. Upon completion, the program documents all the modifications and, if requested, a complete report of the library is provided using the report generation program. Control is then returned to the supervisor program.

The report generation program serves to document the contents of the library (as indicated above) or the contents of the generated program. The purpose of the documentation is to provide hard copy for reference on a particular program or for the inclusion in a technical report. As a convenience, the format of the report fits upon the standard size paper (8½ x 11 inches). A typical output is illustrated by the generated subroutine FG in Appendix B.

The third option of the GCP is the generation of a source program in FORTRAN IV language. As indicated in Fig. 1, the system can generate three distinct types of programs: equilibrium chemistry, finite rate chemistry, and the linearized form of the finite rate equations. In this report, only the latter option is of primary interest, but for completeness, a brief description of the other two options is included.

The equilibrium chemistry generator of the GCP is used to automatically develop subroutines which will compute the equilibrium state of a chemistry system given the pressure and the temperature. The equilibrium state is calculated by minimizing the Gibbs Free Energy function. The minimization is accomplished by numerical iteration using the Newton-Raphson method in n variables. This method is readily extended to the pressure-entropy problems in a direct way as described in Ref. 2.

The finite rate chemistry option is intended to generate programs which evaluate in an exact manner the right side of Eq. (3.1) for a specified set of species and reactions. By specifying a small number of inputs (e.g. the species and reaction numbers as they appear on the library tape), the system will produce the FORTRAN IV source program, which will include subroutines for computing the mixture enthalpy

$$h = \sum_i \alpha_i h_i(T)$$

where α_i is the i^{th} mass fraction, and h_i is the polynomial fit of enthalpy for the i^{th} species. The coefficients of the polynomial h_i are taken directly from the library tape of the GCP. In addition, this portion of the system also generates a subroutine which will compute the temperature given the enthalpy. Numerous other functions can be automatically generated under this option. and these definitions and details can be found in Ref. 1.

The third type of program generated is the linearized form of Eq. (3.1). A source program in FORTRAN IV is automatically developed to evaluate the elements of the A matrix as defined by Eq. (3.3). Further, a vector with elements $\dot{y}_i$ is also generated, and this vector and the elements a_{ij} of the A matrix can then be used by the calling program to compute the vector b in Eq. (3.4). The method used to generate the program is described in Section 3.1 and samples of a typical generated program can be found in Appendix B.

3.2.2 Input Format for the Generation of the Linearized Source Program

This section describes the input format for the linearized option of the GCP. In the following table of card formats, the cards numbered 1 through 9 are required for the linearized option. Card number 3 is used to specify whether the generated routine is to include the logic for iteration on temperature. The cards 10 through 14 are used to encode a subroutine to compute:

(1) Mixture enthalpy given temperature and species mass fractions.

(2) Temperature given mixture enthalpy and species mass fraction.

This subroutine, which is not specifically a part of the linearized subroutine, is usually required by the main program, and for completeness the method of obtaining this routine is indicated. An example of the input cards for eleven species air can be found in the rear of Appendix B.

Under the column describing the card inputs, the symbols between quotation marks are to be transcribed directly to the card.

Card No.	Columns	Format	Description	
1	3-4	I2	month	date
	5-6	I2	day	
	7-8	I2	year	
	45	I1	"1"	
	46	I1	"1"	
2	1	I1	"7"	
	41-45	I5	ns = number of species to be included ≤ 100	
	46-50	I5	nr = number of reactions ≤ 100	
3	1	I1	"3"	
	6	I1	"0" if temperature iteration is desired, else "1"	
4,4′,4″	1-72	6A12	Chemical symbols of the species to be included are punched six species per card. Each species is allocated 12 columns. The symbols can appear anywhere in the 12 columns. The order of the species on these cards determines the order of the species in the program. Species symbol spelling must agree with the library report.	

Card No.	Columns	Format	Description
5,5′,5″	1-70	14I5	Reaction numbers from the library report of the reactions to be included are punched fourteen reactions per card. The numbers are right adjusted and must be in ascending order.
6-9			Four blank cards
10			Same as card #1
11	1	I1	"7"
	41-45	I5	Same as columns 41-45, card 2
12			Columns 3, 4, 5, 6, 9, 14, 15, 18, 19, 21 contain "1"; all others are blank.
13,13′,13″			Same as cards 4, 4′, 4″
14	7-9	A3	"END"

3.2.3 Operating Instructions for Generating the Linearized Source Program

The set of data cards described in Subsection 3.2.2 is placed between the two end-of-file cards at the back of the GCP running deck. The running deck is then executed on an IBM 7090/94 under version 13 of IBSYS. The instructions to the machine operator are:

(1) Mount the GCP library tape on A5; mount scratch tapes on A6, B5, B6.

(2) Execution time is less than 5 minutes.

(3) When job terminates, print B1 and punch two files from A6.

3.2.4 Output of the GCP

The output from the GCP consists of a printed report of the program and a source program deck in FORTRAN IV. In Appendix B, the generated subroutine for the linearized form is listed under the subprogram FG. This program is for dissociating air consisting of the following eleven species: O, O_2, N, N_2, NO^+, e^-, NO, O^+, O_2, N^+, and N_2^+. The subroutine FT, which is also automatically encoded, is used to compute mixture enthalpy or conversely temperature given mixture enthalpy.

The inputs required to generate subroutines FG and FT are listed at the rear of Appendix B and can be compared with the description under Subsection 3.2.2.

3.3 Streamtube Program

Having developed a computer program which will automatically produce a routine that evaluates the linear approximation to the differential equations of any chemical system, it was found desirable to write a general program that utilizes these routines. A subroutine was written that uses the Pade' method to integrate any system of linear ordinary differential equations. This subroutine is combined with the linearization routine and a routine to obtain the temperature from the given temperature-enthalpy relationships; the combined program is used to calculate the chemical kinetic properties along the streamtube. The program can be adapted to any chemical kinetic system by merely changing the linearization and temperature-enthalpy routines, which are both obtained from the GCP.

The Pade' integration routine used in the program assumes that y(0)=0. This assumption simplifies Eqs. (2.9) and the linearized system is then conveniently expressed in terms of Δy as

$$(I - \frac{1}{2} hA + KA^2)\Delta y = hb \qquad (3.5)$$

where K=0 for a second order integration scheme and $K=h^2/12$ for a fourth order integration scheme. By permitting K to be specified as an input, either the second order or fourth order methods can be conveniently selected. In cases where the higher order scheme offers no great advantages, the elimination of the calculation of the matrix A^2 can save an appreciable amount of computer time. The system (3.5) is solved by the Crout reduction method to obtain Δy.

In addition to the species Eq. (3.1) or its numerical form (3.5), the streamtube calculation requires the following equations

$$\frac{du}{dt} = -\frac{1}{\rho}\frac{dp}{dx} \tag{3.6}$$

$$\frac{dx}{dt} = u \tag{3.7}$$

$$h = H - u^2/2 \tag{3.8}$$

$$h(T) = \sum_i \alpha_i \, h_i(T) \tag{3.9}$$

$$\rho = p/(RT\Sigma\alpha_i/w_i) \tag{3.10}$$

where x is the physical distance along the streamtube and all differential equations have been cast in terms of time, t, as the independent variable. The inputs to the program are the initial mass fractions α_i, velocity u, position x, and either the temperature T or static enthalpy. Depending upon what is initially given, Eq. (3.9), which defines the static enthalpy in terms of the mass fractions and their enthalpies, is solved for the initial temperatures or static enthalpy. The constant stagnation enthalpy H is then determined from Eq. (3.8) and the density ρ from Eq. (3.10). The pressure terms in Eqs. (3.6) and (3.10) are given input functions which for the purpose of this program are assumed to be quadratic polynomials in x over various segments of the streamtube. Equation (3.1) is then linearized and solved for y_i over a given time step. If the density is assumed constant over a step, Eqs. (3.6) and (3.7) represent a linear system for u and x, which is also solved by the Pade' integration procedure. Using the new values of y_i, u, and x, the thermodynamic state can be determined.

An important problem in any integration procedure is the selection or control of stepsize. While it is economically desirable to use as large an interval as possible, the stepsize must not be increased to a point where truncation errors may mask the true solution. Further, the stepsize should be varied from step to step due to the changing characteristics of the chemical system. The first obvious test is to reduce the stepsize wherever negative mass fractions appear. However, this alone is not enough since the mass fraction may be very inaccurate although not negative. The criterion in the streamtube program is to determine stepsize on the basis of temperature change. Experience shows that in regions of relatively small temperature change the solution was fairly linear, while it was strongly non-linear in regions of great temperature change. Hence temperature change could be used as a means of controlling stepsize. If the temperature change becomes too large the stepsize is cut in half and if the temperature change is small the stepsize is doubled. This method was used very successfully with dissociating air systems.

Another important consideration in the streamtube program is the effect of retaining only first order species terms in the numerical method. If Eq. (3.1) is written in the general form $\dot{y} = f(y, T, \rho)$ and expanded through first order terms, the linearized equation becomes

$$y = f(y_o, T_o, \rho_o) + (\partial f/\partial y)_o \, \Delta y + (\partial f/\partial T)_o \, \Delta T + (\partial f/\partial \rho)_o \, \Delta \rho \tag{3.11}$$

where y, f, $\partial f/\partial T$ and $\partial f/\partial \rho$ are vectors, $\partial f/\partial y$ is a matrix and $(\ )_o$ implies the function is evaluated at the beginning of the step. The density term can be eliminated in terms of T and y by introducing the equation of state. However, experience has indicated that the density term in (3.11) was of minor importance in the numerical solution for a variety of chemical systems. Hence the entire $\Delta\rho$-term is eliminated from (3.11). The method of handling the temperature term is not straightforward and several procedures have been utilized. For some chemistry systems, the change in temperature over an integration step is sufficiently small, and the ΔT-term can be eliminated without any significant effect upon the numerical solution. Equation (3.11) then simplifies to

$$y = f(y_o, T_o, \rho_o) + \left(\frac{\partial f}{\partial y}\right)_o \Delta y \qquad (3.12)$$

But this simplification is only satisfactory for some air chemistry systems when very small stepsizes are used. To overcome this difficulty, a program was developed to include the term $(\partial f/\partial T)_o \, \Delta T$ in (3.11). Initially ΔT is unknown and the program has been designed to iterate until the solution converges on the correct value. As mentioned under Subsection 3.1, the generator program has the option of producing programs with or without the added temperature term. For those chemistry systems which do not involve large changes in temperature over an integration step, the user can select the simpler option and avoid the added computation associated with iteration. For air chemistry systems as illustrated by Appendix B, the iteration option should be included.

4. RESULTS FROM STREAMTUBE CALCULATIONS

While the techniques of numerical analysis have been used extensively to examine truncation error and obtain indications of the behavior of the propagated error, it is not possible to accurately predict the error in a numerical solution for a given problem. In some cases, a bound on the solution error can be derived, but unfortunately these bounds are much too conservative to provide useful information. Consequently, the solution error, which is an important consideration in the application of the Pade' integration procedure, has been examined via numerical experimentation.

Several typical fluid flow problems involving various chemical kinetics systems have been investigated using the Pade' procedure, and the results have been compared with the solution obtained by other numerical methods. Two of these problems are presented herein. For both problems, the Runge-Kutta integration procedure has been used for comparison purposes. Since the Runge-Kutta procedure is limited with respect to its stepsize, the numerical solution is initially obtained using a stepsize which is considerably less than predicted by the stability analysis, and then the problem is rerun using a stepsize very close to the stability limit. In this manner a very accurate solution is obtained for the purposes of the comparison with the results from the Pade' procedure. The second run of the Runge-Kutta procedure is used to measure the minimum amount of machine time to obtain the solution. In this way, the saving in machine time through the use of the Pade' procedure is demonstrated without any built-in penalty against the Runge-Kutta method.

The first problem to be considered is a streamtube calculation for seven species air consisting of O_2, N_2, O, N, NO, NO^+, and e^-. The initial conditions and the polynomials describing the variation of the pressure are given in Table I. The pressure polynomials which apply over discrete intervals are expressed in terms of s, the nondimensional distance along the streamtube.

In Fig. 2, the variation of temperature and velocity along the streamtube is shown. The solid curves are results obtained using the Runge-Kutta integration procedure whereas the symboled points are from the Pade' numerical solution.

In Figs. 3 and 4, the behavior of the species α_{O_2} and α_{e^-} are shown. Initially the species α_{O_2} decreases rapidly from its value at x = 0. In contrast, the mass fraction of electrons initially increases to a value of approximately 4.25 x 10^{-8} and then starts decreasing. The increase in electrons occurs over the first few steps and represents an adjustment in the initial condition ($\alpha_{e^-} \to 0$) which is inconsistent with the specified temperature.

In all of these figures, the results from the Pade' integration procedure are in good agreement with the results from the Runge-Kutta procedure. Regarding the amount of computer time, the Pade' procedure required approximately 8 sec on the IBM 7094 to integrate over the 0.5 ft domain, whereas the Runge-Kutta method took 200 sec. Hence for this chemistry system the running time is reduced by a factor of about 25.

As a second example, consider the chemistry system for the species O_2, N_2, O, N, NO, NO^+, e^-, O^+, O_2^+, N^+, and N_2^+. Again a streamtube problem was run using the Runge-Kutta and the Pade' integration procedures. The initial conditions and the pressure polynomials are given in Table II.

In Fig. 5, the temperature and velocity results are presented, and the results from the Runge-Kutta (solid line) and the Pade' (symboled points) procedures are in good agreement. Similarly the species results from the two numerical procedures are in satisfactory agreement (Figs. 6 and 7).

For tne mass fraction α_{O_2} (Fig. 6) the scale on the ordinate has been greatly enlarged to illustrate the behavior of the Runge-Kutta procedure when an integration step is slightly too large. The value of the mass fraction (at x = .16) has an oscillatory behavior and if the stepsize is not immediately reduced the amplitude of the oscillation would rapidly grow in an unbounded manner. At such a point, some integration routines cut the stepsize and restart the integration procedure at an earlier location where the solution was stable. For the purpose of this study, a simpler approach was used; namely, the integration routine detects the instability, reduces the stepsize, and continues the integration.

For this problem, it was convenient to make the comparison for computer time over the streamtube interval (0, 0.2) ft. The Pade' procedure required 28 sec whereas the Runge-Kutta about 650 sec. Extrapolating these results to the full length of the streamtube calculation (.5 ft), the running times would be 70 sec and 25 min for the Pade' and Runge-Kutta methods, respectively. The example serves to illustrate the importance of the rational approximation method in reducing the amount of computer time for streamtube calculations.

TABLE I

INITIAL CONDITIONS AND PRESSURE VARIATION FOR THE SEVEN SPECIES AIR PROBLEM

$T = 16836\ ^\circ K$

$U = 6524$ ft/sec

$\alpha_{O_2} = .2378$

$\alpha_{N_2} = .7622$

$\alpha_O = \alpha_N = \alpha_{NO} = \alpha_{NO^+} = \alpha_{e^-} = 0$

Pressure Polynomials

$$p = a_o + a_1 s + a_2 s^2 \quad (lb/ft^2)$$

where $s = x/\ell$, x is the distance along the streamline and $\ell = 0.5$ ft, the nose radius.

a_o	a_1	a_2	Domain of s for Polynomial Fit Start	 End
5961.1	-3694.7	0	0	0.19059
6817.1	-8186.1	0	0.19059	0.551
6181.1	-9100.2	3753.9	0.551	0.681
6344.7	-9318.3	3721.4	0.681	0.8168
6102.0	-8699.7	3326.6	0.8168	1.0

TABLE II

INITIAL CONDITIONS AND PRESSURE VARIATION FOR THE ELEVEN SPECIES AIR PROBLEM

$T = 22806\ ^\circ K$

$U = 7561$ ft/sec

$\alpha_{O_2} = .232$

$\alpha_{N_2} = .768$

$$\alpha_{O} = \alpha_{N} = \alpha_{NO} = \alpha_{NO^+} = \alpha_{e^-} = \alpha_{O^+} = \alpha_{O_2^+} = \alpha_{N^+} = \alpha_{N_2^+} = 0$$

Pressure Polynomials

(See Table I for definition of Polynomial form)

a_0	a_1	a_2	Domain of s for Polynomial Fit	
			Start	End
9671.3	- 5994.3	0	0	0.19059
11060.1	-13280.2	0	0.19059	0.551
10028.2	-14764.1	8090.4	0.551	0.681
10293.7	-15118.1	6037.6	0.681	0.8168
9901.2	-14114.4	5397.1	0.8168	1.0

5. DISCUSSION OF NUMERICAL METHODS FOR PARTIAL DIFFERENTIAL EQUATIONS

It was shown in Section 2 that standard methods for integrating ODE place a restriction on the stepsize. Unless the stepsize is kept relatively small, the numerical solution will become unstable. As shown below, similar instabilities arise in the numerical solution of partial differential equations. However, a much less severe restriction upon stepsize can be obtained by suitably transforming the equations and then applying the methods of Section 2.

For standard finite difference methods the numerical solution of a parabolic partial differential equation will not be stable unless certain restrictions are placed on the mesh size. Consider an equation of the form:

$$\alpha_t(x,t) = a\,\alpha_{xx}(x,t) - b\,\alpha(x,t)$$

$$\alpha(x,o) = f(x) \qquad (5.1)$$

$$a,b > 0$$

where $f(x)$ and the solution $\alpha(x,t)$ are bounded. Let the numerical solution be denoted by $A(x,t)$. Place a uniform mesh size of h in the x direction and K in the t direction with lattice points (x_i, t_j) where $x_i = ih$; $i = 0, \pm1, \pm2 \ldots$, and $t_j = jK$; $j = 0, 1, 2 \ldots$ Further set $A(x_i, t_j) = A_{i,j}$. Using forward and central differences the numerical solution must satisfy the finite difference equation

$$\frac{1}{K}\,(A_{i,j+1} - A_{i,j}) = \frac{a}{h^2}\,(A_{i+1,j} - 2\,A_{i,j} + A_{i+j}) - bA_{i,j}$$

or

$$A_{i,j+1} = \lambda a\, A_{i+1,j} + (1 - 2\,\lambda a - Kb)A_{i,j} + \lambda a\, A_{i-1,j} \qquad (5.2)$$

where $\lambda = K/h^2$. Starting with $A_{i,o} = f_i$, (5.2) can be solved for $A_{i,j+1}$ knowing $A_{i,j}$. However, if $Kb > 2$ then since $a > 0$

$$1 - 2\lambda a - Kb < 0 \qquad (5.3)$$

and an error introduced into the numerical solution will not remain bounded. To illustrate the numerical behavior let

$$A_{i,o} = \begin{cases} \epsilon; & i = 0 \\ 0; & i \neq 0 \end{cases}$$

The solution of (5.2) corresponding to this initial condition will show how a single error introduced at i=0 propagates. First note that if A is non-zero at any two adjacent points it differs in sign at these points. This is obvious for all i at j=0 and j=1. From (5.2) and (5.3) it follows immediately that if $A_{i,j}$ has one sign then $A_{i,j+1}$ will have the opposite sign. Hence

$$\left|A_{i,j+1}\right| = \lambda a \left|A_{i+1,j}\right| + (2\lambda a + Kb - 1)\left|A_{i,j}\right| + \lambda a \left|A_{i-1,j}\right|$$

Let

$$S_j = \sum_i \left|A_{i,j}\right|$$

then

$$S_{j+1} = (4\lambda a + Kb - 1)S_j = (4\lambda a + Kb - 1)^{j+1}\,\epsilon$$

For any j at most 2j+1 terms are non-zero. Suppose the term of largest magnitude is $A_{n,j}$. Then $S_j \le (2j + 1)\left|A_{n,j}\right|$ and $\left|A_{n,j}\right| > (4\lambda a + Kb - 1)^j\,\epsilon/(2j + 1)$. But, if $Kb > 2$, $4\lambda a + Kb - 1 > 1$. Hence, the initial error, ϵ, is amplified from step to step and the solution is unbounded. For the numerical method to remain stable, the necessary condition is

$$\left|1 - 4\lambda a - Kb\right| < 1$$

or

$$K < \frac{2h^2}{4a+h^2 b} \qquad (5.4)$$

If $h^2 b \ll a$, the well known stability condition for the simple diffusion problem follows from (5.4). However, in the usual finite rate problem, the b term, which represents the species generation, is very large compared to a. Consequently, the stability condition is approximately

$$K < \frac{2}{b} \tag{5.5}$$

This limitation upon the stepsize is the same form as discussed in Section 2 for ordinary differential equations using the Euler method. For a large value of b, the stepsize is severely restricted.

The above result has been derived using an explicit type of difference formulation for (5.1), and it is quite natural to try and overcome the restriction with an implicit differencing method. For the simple diffusion problem (b=0), the method is known to be unconditionally stable and by a direct extension, the method can be shown to be stable when $b \neq 0$. However, the simple form of Eq. (5.1) is not representative of species conservation equations when implicit numerical methods are used. The species generation terms, which have the form of the right side of Eq. (3.1), are non-linear functions of all the dependent variables. Within a suitable approximation, these terms can be linearized as described in Section 2, and then each species conservation equation is linear but still coupled with every other species. Specifically, (5.1) becomes

$$\alpha_t^{(r)}(x,t) = a\,\alpha_{xx}^{(r)}(x,t) - \sum_{k=1}^{n} b^{(k)}\alpha^{(k)}(x,t) - c^{(r)} \tag{5.6}$$

where the superscript r is an index ranging over the number of species and the coefficients $b^{(k)}$ and $c^{(r)}$ are obtained from the linearized form of (3.1). The implicit method of solution of (5.6) can be applied in several ways. For example, if the integration is from t_j to t_{j+1}, the difference scheme can be written as

$$- a\lambda\, A_{i-1,j+1}^{(r)} + (1 + b^{(r)}K + 2a\lambda)A_{i,j+1}^{(r)} - a\lambda\, A_{i+1,j+1}^{(r)}$$
$$+ \sum_{k\neq r} Kb^{(k)}A_{i,j+1}^{(k)} = A_{i,j}^{(r)} - Kc^{(r)} \tag{5.7}$$

If there are n species and i lattice points in the x direction the system of simultaneous equations involves n·i unknowns. The coefficient matrix is tridiagonal along the main diagonal with n-1 additional diagonals spaced every i columns. Hence the matrix has numerous zero elements. Although algorithms could be developed to efficiently solve the systems of equations, the number of machine operations would still be large, and since the calculations must be repeated at each integration step, the amount of machine time may become excessive. Furthermore, the amplification matrix, which is the inverse of the matrix represented by the left side of (5.7), may only be convergent with a small stepsize. Unlike the usual diffusion problem, the off-diagonal terms $b^{(k)}$ $(k\neq j)$ can be very large, and the stability behavior of the implicit procedure is completely altered. Therefore, this numerical procedure is not considered satisfactory for the efficient integration of the reaction equations. This conclusion has also been verified by numerical experimentation with small chemical kinetic systems.

A second method of implementing the implicit method is to iterate for the terms involving the $b^{(k)}$ coefficients. The difference equation becomes:

$$- a\lambda\, A^{(r)}_{i-1,j+1} + (1 + Kb^{(r)} + 2\lambda a)A^{(r)}_{i,j+1} - a\lambda\, A^{(r)}_{i+1,j+1}$$

$$= A^{(r)}_{i,j} - Kc^{(r)} - \sum_{k\neq r} Kb^{(k)}A^{(k)}_{i,j+1} \qquad (5.8)$$

and the matrix is then tridiagonal and easily inverted. The $A_{i,j+1}$ terms on the right side are initially evaluated using the numerical results from the previous station; i.e. A_{ij}. Subsequent iterations utilize the previously iterated value of $A_{i,j+1}$. Unfortunately, this iteration procedure is convergent only, if the stepsize K is small; again, the cause of this difficulty is the large value of $b^{(k)}$.

To overcome the numerical difficulties that accompany the standard explicit and implicit numerical methods, a technique (see Ref. 6) has evolved wherein the diffusion and species generation terms are handled independently. The generation terms appear as ordinary differential equations, and the diffusion terms are separate partial differential equations. The solutions are then combined at the end of each integration step. Although the procedure imposes rather strong assumptions regarding the species behavior over a step, a substantial increase in the stepsize is possible. For this method, let $\alpha = p + q$, where p and q are defined between t_j and t_{j+1} by

$$p_t(x,t) = a\,\alpha_{xx}(x,t) \tag{5.8}$$

$$q_t(x,t) = -\,b\,\alpha(x,t) \tag{5.9}$$

In order to solve (5.9) at $x = x_i$, assume the p term is constant between t_i and t_{j+1}; that is, $p(x_i,t) \approx p_{i,j}$. Then in (5.9) the only independent variable is t, and

$$q_i'(t) = - b\, q_i(t) - b\, p_{i,j} \tag{5.10}$$

$$q_i(0) = q_{i,j} = 0$$

The solution of (5.10) is

$$q_i(t) = p_{i,j}(e^{-bt} - 1)$$

Equation (5.8) can be solved directly by the explicit finite difference method

$$p_{i,j+1} = \lambda a\, p_{i+1,j} + (1 - 2\lambda a)p_{i,j} + \lambda a\, p_{i-1,j}$$

By combining these two solutions, the consistent finite difference form is

$$A_{i,j+1} = \lambda a\, A_{i+1,j} + (e^{-Kb} - 2\lambda a)A_{i,j} + \lambda a\, A_{i-1,j} \tag{5.11}$$

Following the same stability argument as before, if

$$2\lambda a > 1 > e^{-Kb} \rightarrow e^{-Kb} - 2\lambda a < 0$$

and for every j there is a point (x_n, t_j) where

$$A_{n,j} \geq (4\lambda a - e^{-Kb})^j\ \varepsilon/(2j+1) .$$

Since $4\lambda a - e^{-Kb} > 1$, an error committed at any step will grow. For the system to remain stable

$$\lambda < 1/2a$$

or (5.12)

$$K < h^2/2a$$

This criterion, which corresponds to the standard diffusion condition, is much less restrictive than (5.5) since b >> a. Thus if the ordinary differential eq. (5.9) can be solved accurately using a large step, the numerical method can overcome the restriction imposed by the large value of b. While standard methods for numerically solving the ordinary differential eq. (5.9) become unstable with a large stepsize, the Pade' integration method developed in Section 2 will yield the desired results. The previous stability analysis is valid with e^{-Kb} replaced by $Q^{-1}(-Kb)\ P(-Kb)$. Since

$$\left| Q^{-1}(-Kb)\ P(-Kb) \right| < 1$$

the overall method remains stable within the restriction of (5.12).

The integration procedure implied by Eq. (5.11) permits a significantly larger stepsize than the standard explicit or implicit methods, and it has been successfully applied to several two-dimensional problems. The saving in machine time for these problems was very significant and further savings are expected as work continues in the study and application of the method. A problem of particular interest is the relationship between the stable stepsize prescribed by (5.12) and the limitation imposed by the truncation arror. Under some circumstances, a fraction of the stable stepsize must be used to achieve the desired accuracy in the numerical solution.

6. CONCLUSIONS AND RECOMMENDATIONS

The limitations on the stepsize for the numerical integration of the finite rate equations has been investigated and explained in terms of the eigenvalues of the amplification matrix. For standard numerical methods of integration (Runge-Kutta, Predictor-Corrector, etc.) the restriction is severe and causes an excessive use of machine time. To overcome the difficulty, an integration procedure which utilizes a rational approximation to replace the exponential term is recommended. This method is unconditionally stable, and the only restriction upon the stepsize is imposed by the usual need for control of truncation and round-off errors which are common to all numerical approximations.

The method is directly applicable to linear ordinary differential equations and is adapted to the nonlinear finite rate equations by a Taylor's Series expansion in n-variables. Since such an expansion is algebraically difficult, special algorithms have been developed and programmed to automatically perform the expansion and produce the required digital computer subroutines. The generator program is completely operational and can be used to develop subroutines for generic chemical systems.

The numerical procedure has been used in the analysis of numerous chemical kinetics systems including systems with as many as 31 species and 78 reactions. For an air chemistry system consisting of 11 species, the rational approximation method of integration used approximately 1/20 the machine time required by the Runge-Kutta procedure. Hence, for a typical stream tube problem, the machine time was reduced from 25 minutes to 70 seconds.

The method has been investigated for application to partial differential equations which can not be satisfactorily integrated using either the standard explicit or implicit methods because of severe stepsize restrictions. Two-dimensional flow problems involving chemical kinetics are of this type. The method has been applied to these problems and satisfactory results have been obtained. However, future research effort should be directed toward further study of the method in this application. In particular, the effect and control of truncation error should be studied in detail with a view toward developing criteria for the control of the integration step.

REFERENCES

1. Hoffman, J. R. and Wecker, M. S., "Program Generator for Finite-Rate Chemical Systems," GASL TR-575, March 1, 1966.

2. Hopf, H. H., "Analysis and Description of an IBM 7090/94 Program to Compute Equilibrium Conditions for Gaseous Chemistry Systems," GASL TR-643, December 14, 1966.

3. Magnus, D. E. and Schechter, H. S., "Analysis of Error Growth and Stability for the Numerical Integration of the Equations of Chemical Kinetics," GASL TR-607, June 1966.

4. Emanuel, G., "Numerical Analysis of Stiff Equations," SSD-TDR-63-380 (AD 431250), January 1964.

5. Varga, R. S., "On Higher Order Stable Implicit Methods for Solving Parabolic Partial Differential Equations," J. Math. Physics, Vol. 40, pp. 220-331

6. Slutsky, S., "Stable Computation Techniques of Coupled Diffusion and Chemical Reaction in Shear Flows," GASL TM-101, December 1963.

7. Curtiss, C. F. and Hirschfelder, J. O., "Integration of Stiff Equations," Nat. Academy of Sciences Proceedings, Vol. 38, p. 235, 1952.

8. Treanor, G. E., "A Method for the Numerical Integration of Coupled First Order Differential Equations with Greatly Different Time Constants," Cornell Aero. Lab., Report No. AG-1729-A-4, January 1964.

9. Sherman, M. P., "Radiation-Coupled Chemical Nonequilibrium Normal Shock Waves," General Electric, R66SD17, March 1966.

10. Centaine, J., "The Solution of Ordinary Differential Equations with Large Time Constants," Mathematical Methods for Digital Computers, John Wiley and Son, Inc., 1960.

11. Emanuel, G., "Problems Underlying the Numerical Integration of the Chemical and Vibrational Rate Equations in a Near-Equilibrium Flow," AEDC-TDR-63-82, March 1963.

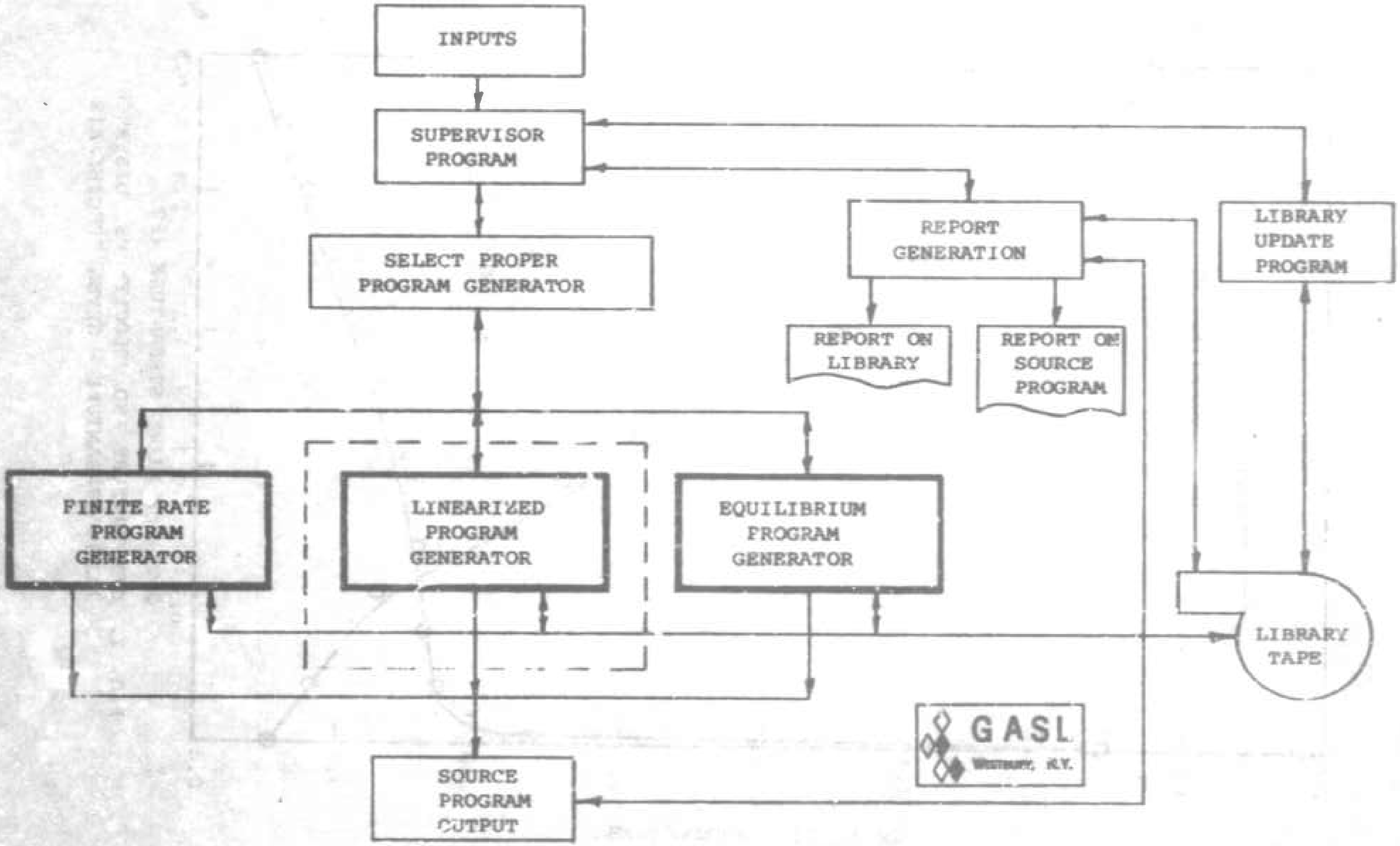


FIG. 1. BLOCK DIAGRAM FOR THE GENERALIZED CHEMISTRY PROGRAM

(Element added to system under this effort is enclosed in the dashed box)

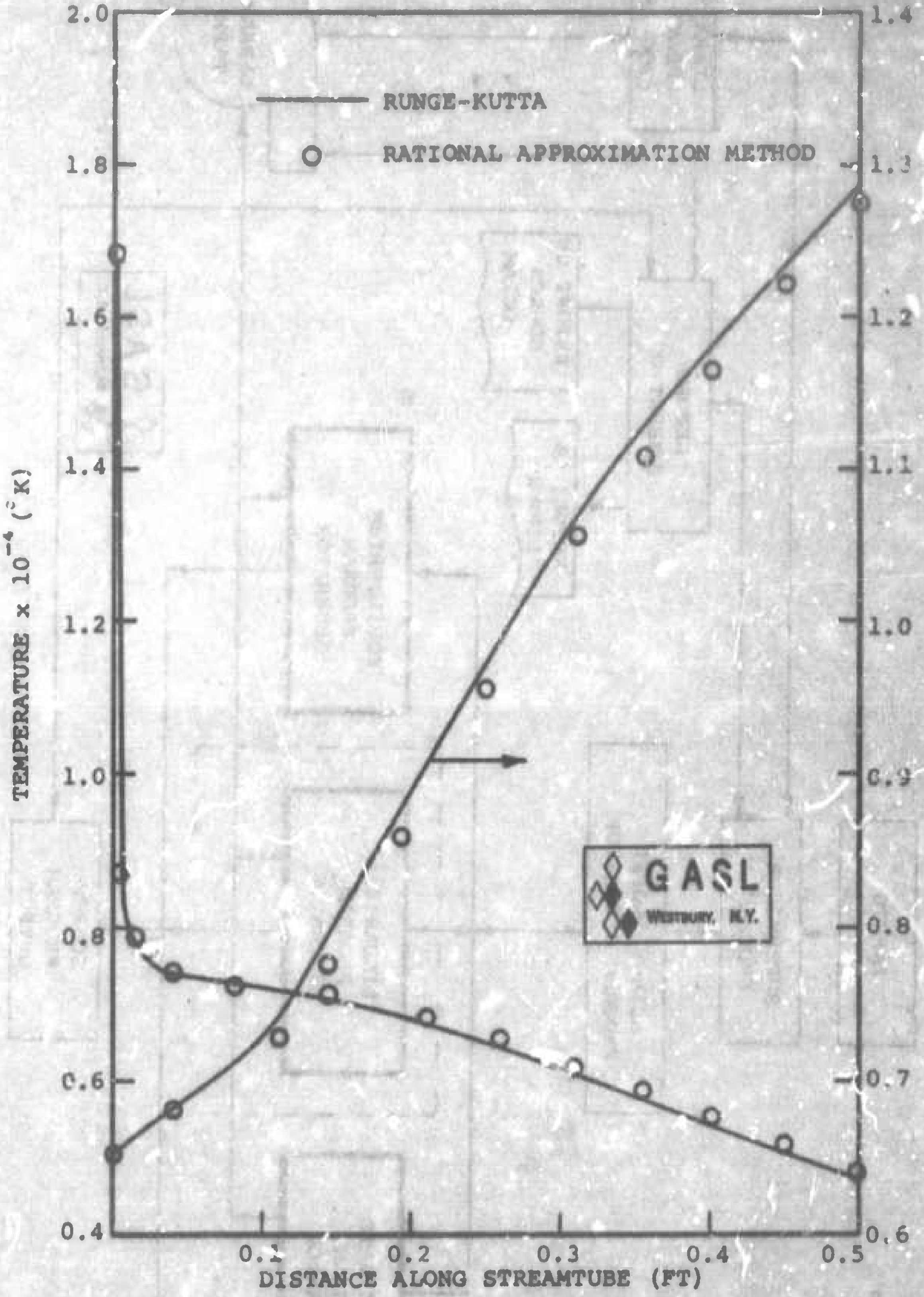


FIG. 2. TEMPERATURE AND VELOCITY VS. DISTANCE ALONG STREAMTUBE - SEVEN SPECIES AIR

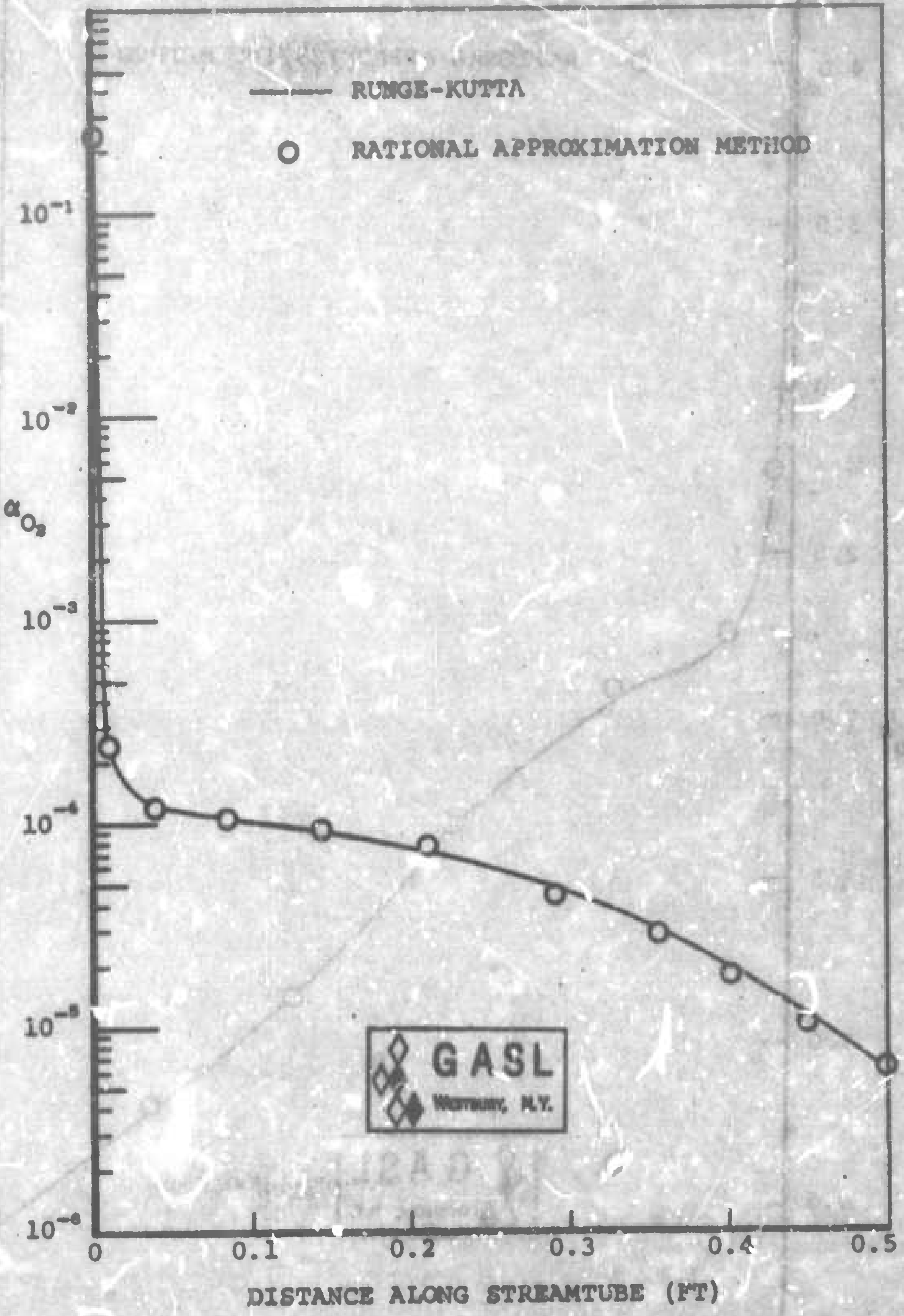


FIG. 3. α_{O_2} VS. DISTANCE ALONG STREAMTUBE - SEVEN SPECIES AIR

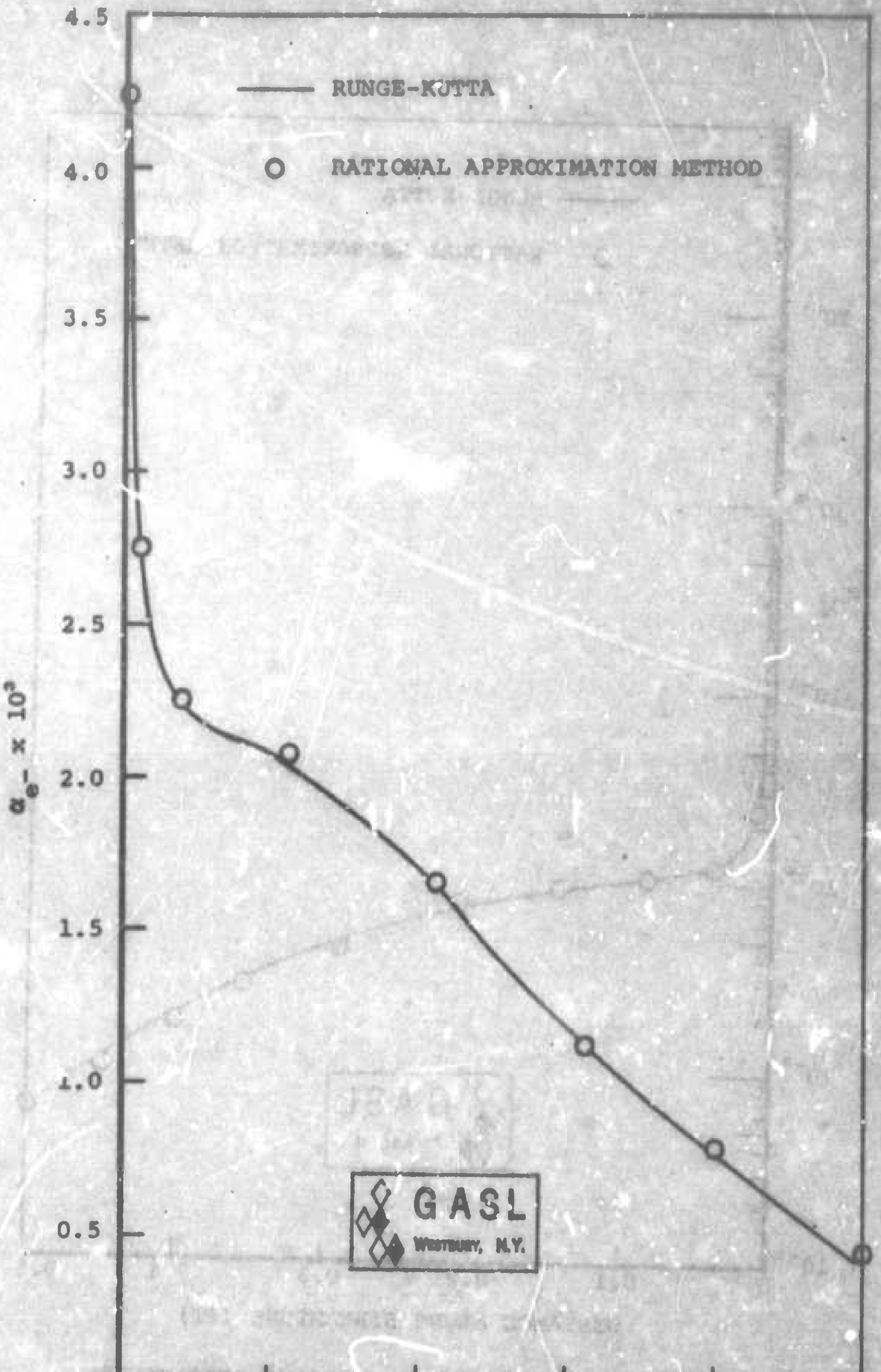


FIG. 4. α_{e^-} VS. DISTANCE ALONG STREAMTUBE - SEVEN SPECIES AIR

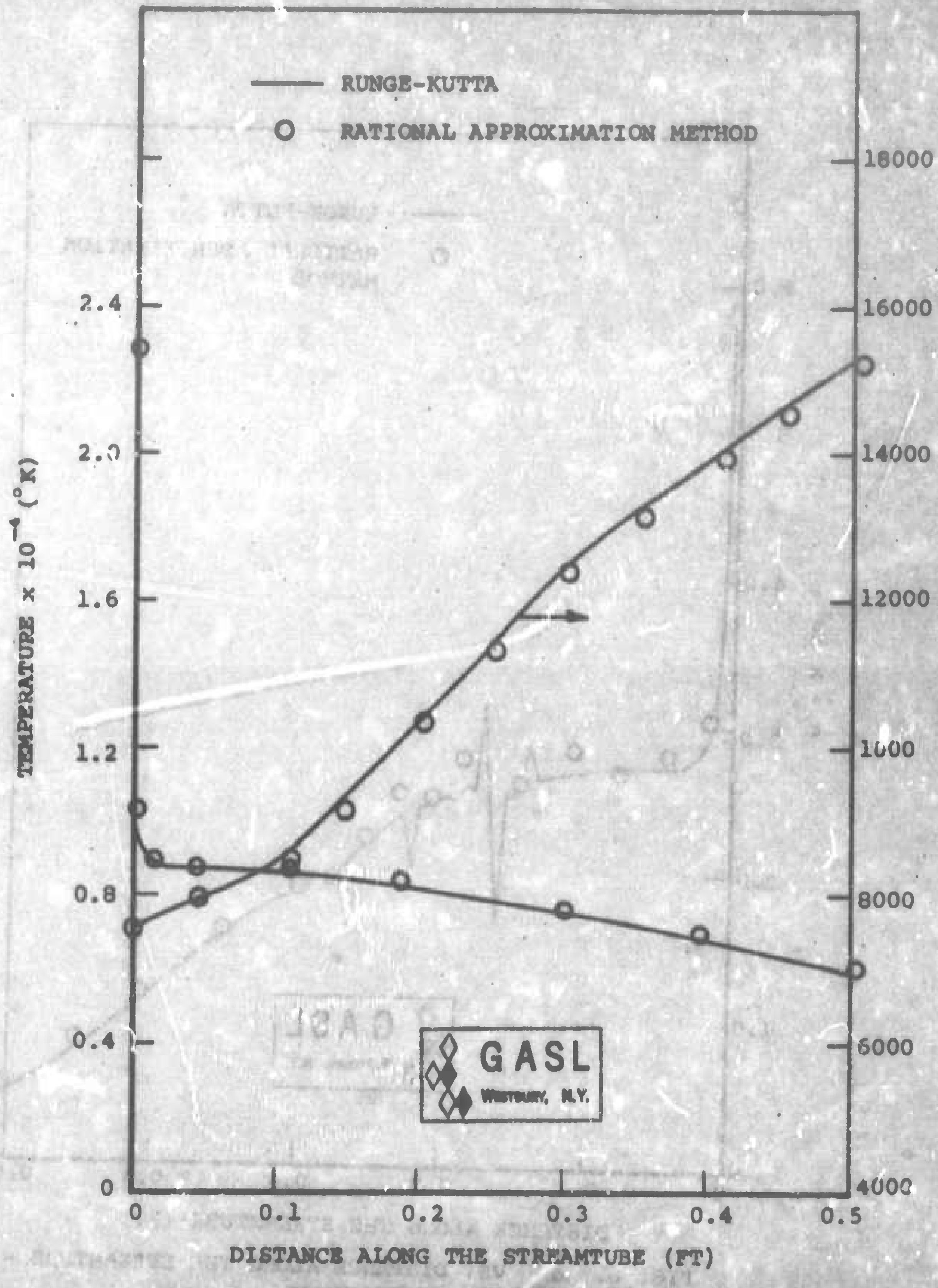


FIG. 5. TEMPERATURE AND VELOCITY VS. DISTANCE ALONG THE STREAMTUBE - ELEVEN SPECIES AIR

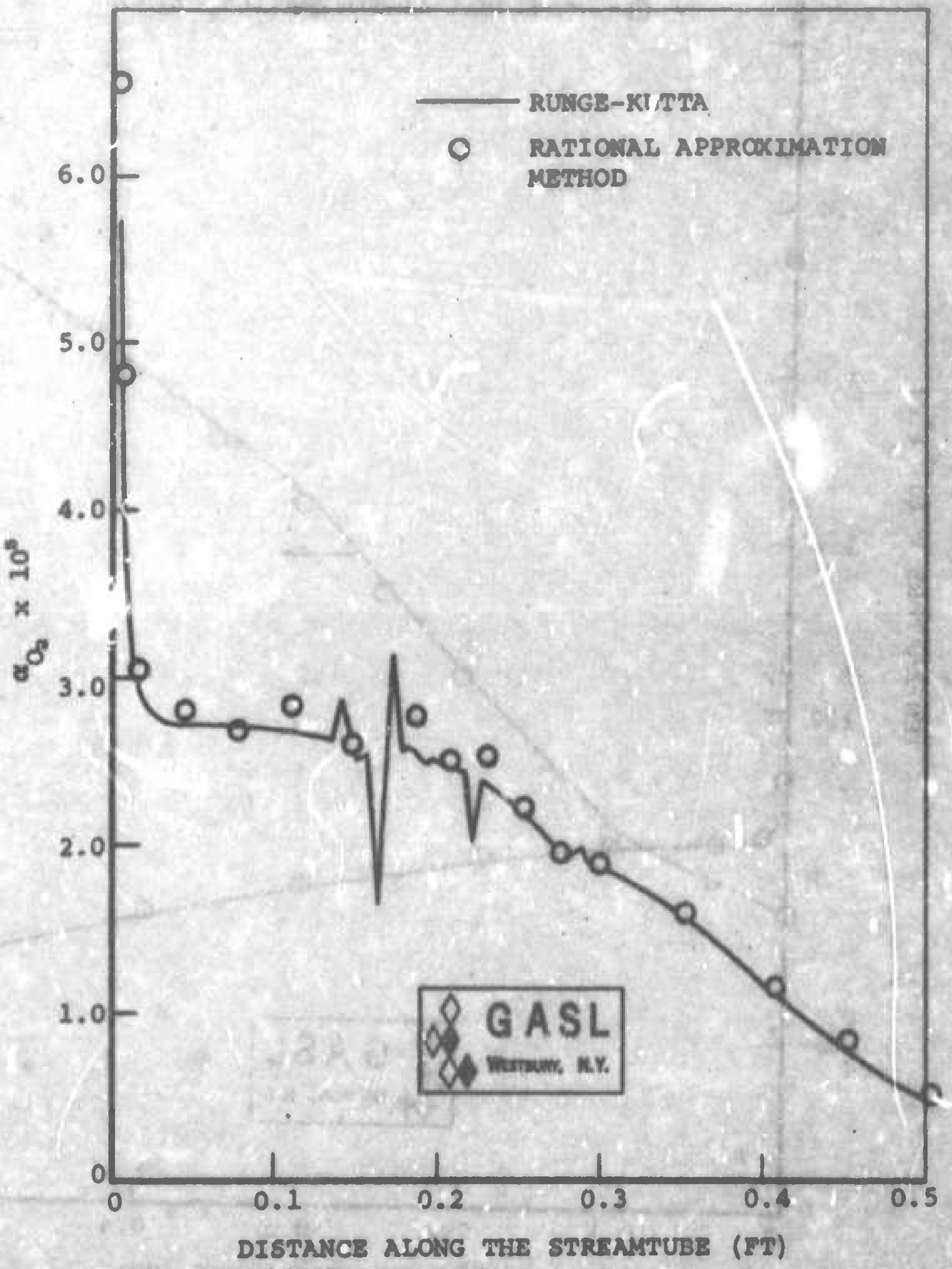


FIG. 6. α_{O_2} VS. DISTANCE ALONG THE STREAMTUBE - ELEVEN SPECIES AIR

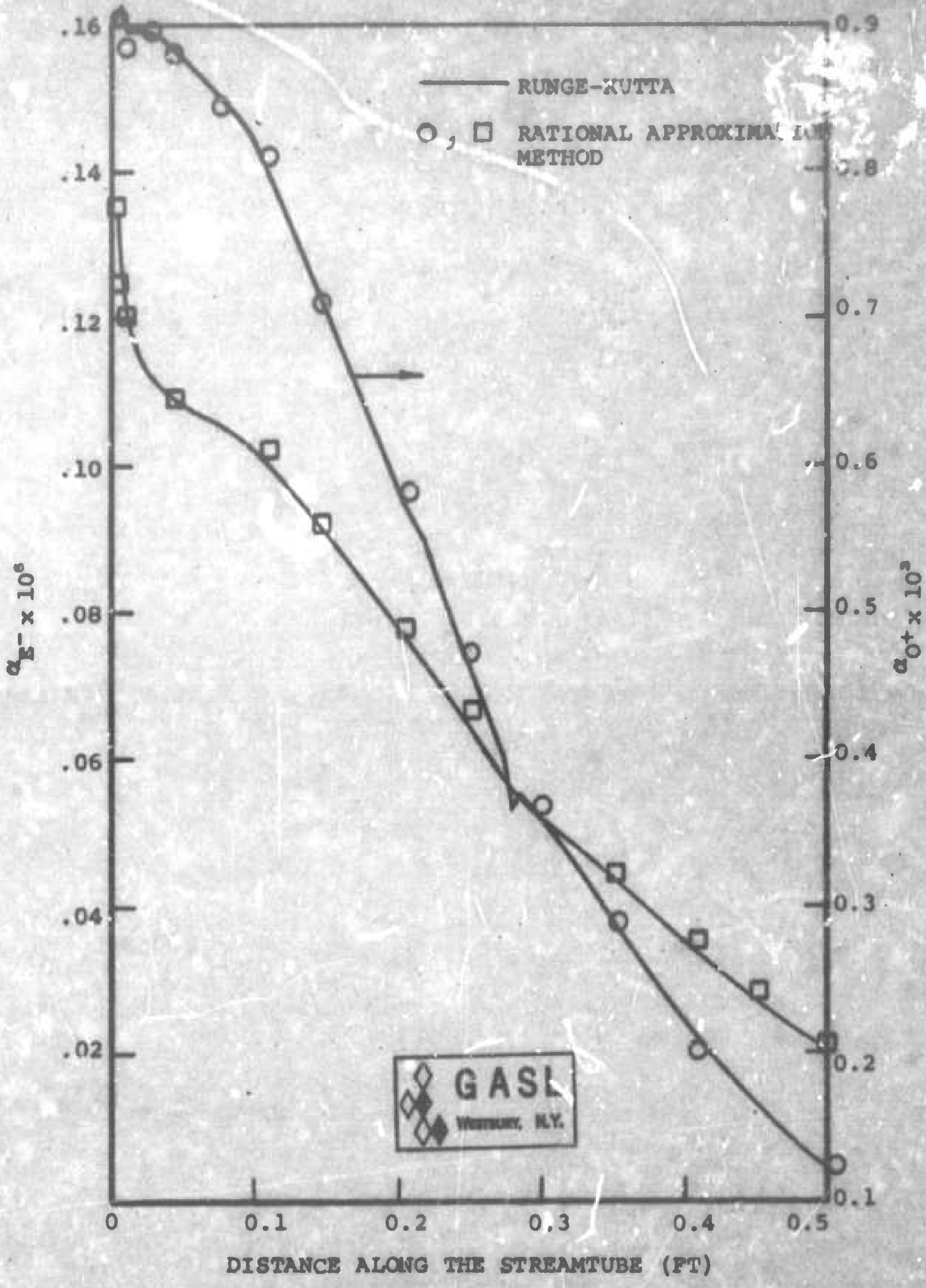


FIG. 7. α_{e^-} AND α_{O^+} VS. DISTANCE ALONG THE STREAMTUBE - ELEVEN SPECIES AIR

APPENDIX "A"

LISTING OF THE GENERATOR PROGRAM

```
$ORIGIN        LINK.1,SYSUT3
$IBFTC HS..LS  LIST
      SUBROUTINE HSXXLS(NN1,NN2,NN3,NN4,NN5)
      COMMON/CNT/IND(100),IB(1003),IBX(1003),A(100)
      EQUIVALENCE(IB,B),(IBX,BX)
      DIMENSION B(1003),BX(1003)
      COMMON/CONST/DATE(3),DES,SETS,SETSX,COL1,PT,IT,IIX(20),KIX(3),NS,N
     1A,NNX,I1,I2,MWT,M1,M2,DESX
      INTEGER DATE,DES,SETS,SETSX,COL1,PT,EQLR,DESX,PTXX,PTXXX
      REAL MWT
      DIMENSION DTX(50,50,3),SYM7X(200,2),WT7X(200),
     1                        RATE7X(6,200),V7X(6300),W7X(6300),ACOD7X(
     212,200),SPEC7X(200),REAC7X(200),NF(3)
      COMMON/NOPTNR/IXY(72)
      EQUIVALENCE(DTX,                  WT7X,RATE7X),(DTX(1201),V7X,W7X),
     4                                           (DES,NSPC7X),(SETS,N
     ZRCT7X)
C             ACOD7X IS NOT    IN /D7X/
      INTEGER SOURCE,POOL,FLCP,SPEC7X,PT,PTX,END,COPY,ACOD7X,REAL7X,V7X,
     1OPTION
      COMMON/D7X/DTX,SYM7X,SPEC7X,REAC7X,NF
      INTEGER EQSYM,POSX
      COMPLEX SHFT,DUM
      DIMENSION JNUM(200,2),SPEC(200,2)
      LONGD=6300
      LIB=NN1
      LIBSOR=NN2
      SOURCE=NN3
      POOL=NN4
      INTAPE=NN5
      REWIND LIB
      REWIND LIBSOR
      REWIND POOL
      READ(LIB)(IB(I),I=1,150)
    6 FORMAT(14I5)
      DO 3 I=1,12
    3 IB(I&19)=IND(I&39)
      DO 5 I=1,72
    5 IB(I&39)=IXY(I)
      IB(150)=3
      WRITE(LIBSOR)(IB(I),I=1,150)
      IF(NRCT7X.GT.200.OR.NSPC7X.GT.200)CALL ERROR(40)
      READ(INTAPE,709)(SPEC(I,1),SPEC(I,2),I=1,NSPC7X)
  709 FORMAT(12A6)
      DO 712 II=1,NSPC7X
      DUM=SHFT(SPEC(II,1),SPEC(II,2),0)
      SPEC(II,1)=REAL(DUM)
  712 SPEC(II,2)=AIMAG(DUM)
      IF(NRCT7X.EQ.0)CALL ERROR(41)
      READ(INTAPE,6)(REAC7X(I),I=1,NRCT7X)
      IF(NRCT7X.EQ.1)GO TO 4
      DO 12 I=2,NRCT7X
      IF(REAC7X(I).LE.REAC7X(I-1))CALL ERROR(36)
   12 CONTINUE
    4 CONTINUE
```

```
  903 READ(LIB)(IB(I),I=1,1003)
      II=1
      III=0
      DO 706 N=1,NSPC7X
  704 IT=POSX(LIB,21,II,4,I6)
      IF(IT.NE.0)CALL ERROR(31)
      PT=(II-IB(1002))*4&1
      DO 703 I=1,NSPC7X
      K=EQSYM(SPEC(I,1),SPEC(I,2),IB(PT&2),IB(PT&3))
      IF(K.EQ.0)GO TO 705
  703 CONTINUE
      II=II&1
      GO TO 704
  705 III=III&1
      SPEC7X(I)=IB(PT)
      JNUM(III,1)=II
      JNUM(III,2)=I
      SYM7X(I,1)=B(PT&2)
      SYM7X(I,2)=B(PT&3)
      WT7X(I)=B(PT&1)
      II=II&1
  706 CONTINUE
      IBX(1001)=21
      IBX(1002)=1
      IBX(1003)=NSPC7X
      PTX=-3
      DO 730 II=1,NSPC7X
      PTX=PTX&4
      IF(PTX.LT.997)GO TO 722
      IBX(1003)=II-1
      WRITE(LIBSOR)(IBX(K),K=1,1003)
      PTX=1
      IBX(1002)=II
      IBX(1003)=NSPC7X
  722 IBX(PTX)=II
      BX(PTX&1)=WT7X(II)
      BX(PTX&2)=SYM7X(II,1)
  730 BX(PTX&3)=SYM7X(II,2)
      WRITE(LIBSOR)(IBX(K),K=1,1003)
      LONGD=6300
      DO 78 I=1,3
   78 NF(I)=0
      DO 79 I=1,NRCT7X
      IF(REAC7X(I).LT.5000)NF(1)=NF(1)&1
      IF(REAC7X(I).GE.5000.AND.REAC7X(I).LT.7000)NF(2)=NF(2)&1
   79 IF(REAC7X(I).GE.7000)NF(3)=NF(3)&1
      NRI=0
      NF127X=NF(1)&NF(2)
      DO 100 II=1,3
      NFX=NF(II)
      IF(NFX.EQ.0)GO TO 86
      NFIRST=1
      IF(II.EQ.2)NFIRST=5000
      IF(II.EQ.3)NFIRST=7000
      DO 80 K=1,1000
   80 BX(K)=0.
      IBX(1001)=II&30
      IBX(1002)=NFIRST
      IBX(1003)=NFIRST-1
      PTX=-24
      DO 85 IK=1,NFX
      NRE=NRE&1
      IT=POSX(LIB,30&II,REAC7X(NRE),25,IB)
      IF(IT.NE.0)CALL ERROR(37)
      PTX=PTX&25
      IF(PTX.LE.976)GO TO 82
      WRITE(LIBSOR)(IBX(K),K=1,1003)
      DO 81 K=1,1000
   81 BX(K)=0.
      IBX(1002)=IBX(1003)&1
      IBX(1003)=IBX(1002)
      PTX=1
      GO TO 83
   82 IBX(1003)=IBX(1003)&1
   83 IBX(PTX)=IBX(1003)
      PT=(REAC7X(NRE)-IB(1002))*25&1
      IF(IB(PT).LT.3)CALL ERROR(44)
      IF(IB(PT).NE.REAC7X(NRE))CALL ERROR(40)
      DO 84 K=1,12
      N=PT&(K-1)*2&1
      M=PTX&(K-1)*2&1
      IBX(M)=IB(N)
      IBX(M&1)=IB(N&1)
      IT=CONV(IB(N),IB(N&1),SYM7X,SPEC7X,NSPC7X)
      IF(IT.LT.0)CALL ERROR(45)
   84 ACOD7X(K,NRE)=IT
   85 CONTINUE
      WRITE(LIBSOR)(IBX(K),K=1,1003)
   36 CONTINUE
  100 CONTINUE
      LONG7X=1
      NRE=0
      DO 130 II=1,3
      NFX=NF(II)
      IF(NFX.EQ.0)GO TO 130
      DO 101 K=1,1000
  101 BX(K)=0.
      IBX(1001)=33&II
      NFIRST=1
      IF(II.EQ.2)NFIRST=5000
      IF(II.EQ.3)NFIRST=7000
      IBX(1002)=NFIRST
      IBX(1003)=NFIRST-1
      PTX=-6
      IF(II.EQ.3)PTX=1
      DO 120 IK=1,NFX
      NRE=NRE&1
      J=7
      IF(II.EQ.3)J=1000
      IT=POSX(LIB,33&II,REAC7X(NRE),J,IB)
      IF(IT.NE.0)CALL ERROR(38)
      IF(II.EQ.3)GO TO 112
      PTX=PTX&7
      IF(PTX.LE.994)GO TO 108
      WRITE(LIBSOR)(IBX(K),K=1,1003)
```

```
      DO 106 K=1,1000
  106 BX(K)=0.
      IBX(1002)=IBX(1003)&1
      IBX(1003)=IBX(1002)
      PTX=1
      GO TO 109
  108 IBX(1003)=IBX(1003)&1
  109 PT=(REAC7X(NRE)-IB(1002))*7&1
      IBX(PTX)=IBX(1003)
      DO 111 K=1,6
      M=PT&K
      N=PTX&K
      IBX(N)=IB(M)
  111 RATE7X(K,NRE)=B(M)
      DO 107 K=1,4,3
      IF(RATE7X(K,NRE).NE.0.)GO TO 104
      RATE7X(K,NRE)=1.E35
      GO TO 107
  104 RATE7X(K,NRE)=ALOG(RATE7X(K,NRE))
  107 CONTINUE
      GO TO 120
  112 M=IB(2)
      DO 113 K=1,M
  113 IBX(K)=IB(K)
      IBX(1)=IBX(1002)
      V7X(LONG7X)=IBX(1002)
      IBX(1003)=IBX(1002)
      LONN=3
      LON=LONG7X&1
      NDE=IB(3)
      V7X(LONG7X&2)=NDE
      LONG7X=LONG7X&3
      IF(LONG7X.GT.LONGD)CALL ERROR(46)
      PT=4
      DO 118 N=1,NDE
      DO 114 K=1,6
      V7X(LONG7X)=IB(PT)
      PT=PT&1
      LONG7X=LONG7X&1
  114 IF(LONG7X.GT.LONGD)CALL ERROR(46)
      DO 116 K=1,4,3
      KK=LONG7X-7&K
      IF(W7X(KK).NE.0.)GO TO 115
      W7X(KK)=1.E35
      GO TO 116
  115 W7X(KK)=ALOG(W7X(KK))
  116 CONTINUE
      LONN=LONN&6
      IF(N.EQ.NDE)GO TO 118
      IS=IB(PT)
      V7X(LONG7X)=IS
      LONN=LONN&1
      LONG7X=LONG7X&1
      IF(LONG7X.GT.LONGD)CALL ERROR(46)
      PT=PT&1
      DO 117 M=1,IS
      IT=CONV(IB(PT),IB(PT&1),SYM7X,SPEC7X,NSPC7X)
      IF(IT.LT.0)CALL ERROR(39)
      V7X(LONG7X)=IT
      LONG7X=LONG7X&1
      LONN=LONN61
      IF(LONG7X.GT.LONGD)CALL ERROR(46)
  117 PT=PT62
  118 CONTINUE
      V7X)LONN)=LONN
      IF(IK.EQ.NFX)GO TO 120
      WRITE(LIBSOR)((BX)K),K=1,1003)
      IAX)1002)=IBX)1002)61
  120 CONTINUE
      WRITE(LIBSOR)((BX)K),K=1,1003)
  130 CONTINUE
      IBX)1001)=99
      WRITE(L)BSOR)((BX)K),K=1,1003)
      CALL AMAKE)RCOD7X,V7X,RATE7X,REAC7X,NRCT7X,NSPC7X,SOURCE,IXY)6),PU
     IOL)
      ENDFILE SOURCE
      RETURN
      END
$.BFTC AMAKE.  LIST
      SUBROUTINE AMAKE)JA,JB,JC,NRLIST,NR,NS,N2,KND,N3)
      DIMENSION JA)12,1),JB(2),NRLIST)2),JC)6,1),JD)6)
      COMMON/DW/I,J,IT,MA,MB,MATEMP(4),MBTEMP)4),INR,JZ,LZ
      COMMON/DS/KK,K,MM
      COMMON/DT/KLA,KLB,IA,IB
      IF(NR.EQ.0.OR.NS.EQ.0)GO TO 1200
      IND=1-KND
C     IND=0   NO ITERATION WITH TEMPERATURE
C     IND=1  ITERATION WITH TEMPERATURE
      REWIND N3
      WRITE)N2,115)
      NRT=0
      II=0
      DO 7 I=1,NR
      IF(NRLIST(I).GE.7000)GO TO 2
      NRT=I
      GO TO 7
    2 IF)II.NE.0)GO TO 4
      II=NRT
      JJ=0
      IF)NRT.EQ.0)GO TO 4
      DO 3 J=1,NAT
    3 WRITE)N3))JC(L,J),L=1,6)
    4 JM=JJ63
      JW=JB)JM)
      JJ=JJ6JB(JJ62)
      II=II6JW
    5 DO 6 J=1,6
      MM=JM6J
    6 JD(J)=JB(MM)
      WRITE(N3)(JD(J),J=1,6)
      JW=JW-1
      IF(JW.EQ.0)GO TO 7
      JM=MM61
      KP=JB(JM)
      JM=JM6KP
```

```
      GO TO 5
    7 CONTINUE
      IF(II.EQ.0)GO TO 8
      NAT=II
      GO TO 10
    8 DO 9 J=1,NRT
    9 WRITE(N3)(JC(L,J),L=1,6)
   10 IF(IND.EQ.0)GO TO 11
      WRITE(N2,110)NS,NS,NS,NR,
      GO TO 12
   11 WRITE(N2,111)NS,NS,NRT
   12 WRITE(N2,112)NRT,NRT
      CALL ODPCH(NRT,N2,N3)
      WRITE(N2,113)NS
      WRITE(N2,121)NRT,NS,NS
      IF(IND.EQ.0)GO TO 21
      WRITE(N2,116)
   21 CALL DERV(JA,JB,NRLIST,NR,N2)
      DO 1000 J=1,NS
      DO 800 K=1,NS
      MM=0
      KLA=0
      KLB=0
      IA=0
      IB=0
      II=0
      DO 500 I=1,NR
      INR=0
      IBETA=NRLIST(I)/7000
      IZETA=NRLIST(I)/3000
      IF(IBETA.EQ.0)GO TO 54
      IF(II.NE.0)GO TO 52
      II=I-1
      JJ=0
   52 INR=II
      JM=JJ63
      JW=JB(JM)
      JQ=JB(JJ62)
      JJ=JJ6JQ
      II=II6JW
      GO TO 55
   54 JW=1
      INR=I
   55 CALL ANUMJK(JA,2)
      JZ=JZ-LZ
      IF(JZ.EQ.0)GO TO 400
      CALL ANUMJK(JA,1)
      KK=0
   56 KK=KK61
      IF(MATEMP(KK).EQ.K)GO TO 57
      IF(KK-MA)56,70,70
   57 DO 65 JRS=1,JW
      INR=INR6IBETA
      CALL STAB(JA,N2,1)
   65 CONTINUE
      INR=IT
      INR=INR-JW*IBETA
   70 KK=0
   71 KK=KK61
      IF(MBTEMP(KK).EQ.K)GO TO 73
      IF(KK-NB)71,80,80
   73 KK=KK&5
      DO 75 JNS=1,JN
      INR=INR&IBETA
      CALL STAB(JA,N2,2)
   75 CONTINUE
      INR=INR&IT
      INR=INR-JN*IBETA
   80 IF(IZETA.EQ.0.OR.INK.EQ.2)GO TO 500
      IJ=1&IBETA
      GO TO (300,400),IJ
  400 DO 410 JQ=1,JN
      IF(JQ.EQ.JN)GO TO 420
      JM=JM&7
      KP=JB(JM)
      JM=JK&KP
      DO 410 JP=1,KP
      NN=JM-(KP-JP)
      IF(JB(NN).EQ.K)GO TO 420
  410 CONTINUE
  420 INR=INR&JQ
  300 DO 320 JQ=1,2
      KK=5*JQ
      CALL STAB(JA,N2,JQ)
  320 CONTINUE
  500 CONTINUE
      NN=1
      IF(IA.EQ.0)GO TO 501
      CALL STAB(JA,N2,3)
  501 IF(IB.EQ.0)GO TO 510
      CALL STAB(JA,N2,4)
  510 IF(KLA.EQ.0.AND.KLB.EQ.0)GO TO 800
      IF(KLA.EQ.0) GO TO 550
      IF(KLB.EQ.0)GO TO 540.
      WRITE(N2,101)J,K
      GU TO 800
  540 WRITE(N2,201)J,K
      GO TO 800
  550 WRITE(N2,301)J,K
  800 CONTINUE
 100C CONTINUE
      CALL MAKE(JA,JB,NRLIST,NR,NS,N2,IND)
      IF(IND.EQ.0)GU TO 1100
      WRITE(N2,118)NRT,NS
      GO TO 1150
 1100 WRITE(N2,119)
 1150 WRITE(N2,120)
 1200 RETURN
  101 FORMAT(6X,2HA(I3,1H,I3,11H) = AA - BB)
  110 FORMAT(6X,37HSUBROUTINE FG)Y,A,D,B,RHOO,TN,L)/6X,12HDIMENSION Y(13
     1,1IH),A(L,L),D(13,4H),B(13,10H),RATES(6,)3,1H))
  111 FORMAT(6X,30HSUBROUTINE FG(Y,A,B,RHOB,TN,L)/6X,12HDIMENSION Y(13,1
     11H),A(L,L),B(13,10H),RATES(6,13,1H))
  112 FORMAT(6X,2HHDIMENSION RHO(6),C)1I),QK))3,7H,2),PL()3,4H,1U))
  113 FORMAT(6X,8HENTRY FF/6X,19HRHO(1)=.515362*RHOB/6X,10HDO 5 I=2,6/6X
```

```
     1,24H5 RHO(I)=RHO(I-1)*RHO)1)/6X,4HX=0./6X,9HDO 6 I=1,13/6X,13HO A=
     2X&Y(I))
  115 FORMAT(10H$IBFTC FG.,3X,4HLIST)
  116 FORMAT(6X,5HIND=0)
  118 FORMAT(6X,15HIF(IND)50,35,50/3X,8H35 IND=1/6X,10HDO 40 I=1,13/6X,29
     1HQ=(RATES(2,I)&RATES(3,I)/T)/T/6X,17HPL(I,1)=Q*PL(I,1)/6X,29HQ=(RA
     2TES(5,I)&RATES(6,I)/T)/T/3X,20H40 PL(I,6)=Q*PL(I,6)/6X,1[illegible] 45 I=
     31,13/3X,12H45 D(I)=B)I)/6X,8HGO TO 30/3X,9H50 RETURN)
  119 FORMAT(6X,6HRETURN)
  120 FORMAT(6X,3HEND)
  121 FORMAT(6X,32HIF(ABS((TN-T)/TN)-.0001)20,20,10/3X,7H10 T=TN/6X,12HT
     1L=ALOG(T)/6X,10HDO 18 I=1,13/6X,12HQ=RATES(1,I)/6X,19HIF(Q-1.E33)1
     22,11,11/3X,13H11 QK(I,1)=0./6X,8HGO TO 13/3X, 33H12 Q=QRATES(2,I)
     3*TL-RATES(3,I)/T/6X,14HQK(I,1)=EXP(Q)/3X,15H13 Q=RATES(4,I)/6X,19H
     4IF(Q-1.E30)15,14,14/3X,13H14 QK(I,2)=0./6X,8HGO TO 15/3X,33H15 Q=Q
     5&RATES(5,I)*TL-RATES(6,I)/T/6X,14HQK(I,2)=EXP)Q)/3X,11H16 CONTINUE
     6/ 3X,13H20 DO 25 I=1,13/6X,10HDO 25 J=1,13/3X,12H25 A(I,J)=0.)
  201 FORMAT(6X,2HA(I3,1H,I3,6H) = AA)
  301 FORMAT(6X,2HA(I3,1H,I3,6H) = - BB)
      END
$IBFTC OOPCH.  LIST
      SUBROUTINE OOPCH(NRT,N2,N3)
      DIMENSION ABT(72),OP(6)
      IB=NRT
      NSIZE=6*IB
      LOOP=(NSIZE-1)/72&1
      NN=72
      JN=12
      LSIZE=-11
      REWIND N3
      DO 50 J=1,LOOP
      LSIZE=LSIZE&JN
      IF(J.EQ.LOOP)GO TO 10
      IB=IB-12
      GO TO 11
   10 JN=IB
      NN=6*IB
   11 KS=0
      DO 13 KK=1,JN
      READ(N3)(OP(I),I=1,6)
      DO 13 I=1,6
      KS=KS&1
      ABT(KS)=OP(I)
   13 CONTINUE
      IF(JN.EQ.1)GO TO 14
      MSIZE=LSIZE&JN-1
      WRITE(N2,2)LSIZE,MSIZE
      GO TO 15
   14 WRITE(N2,6)LSIZE
   15 KK=NN/4
      IF((NN-4*KK).EQ.0)GO TO 30
      NW=NN-2
      WRITE(N2,4)(ABT(I),I=1,NW)
      WRITE(N2,5)ABT(NW&1),ABT(NW&2)
      GO TO 50
   30 NW=NN-4
      WRITE(N2,4)(ABT(I),I=1,NW)
      WRITE(N2,7)ABT(NW&1),ABT(NW&2),ABT(NW&3),ABT)NW&4)
   50 CONTINUE
      RETURN
```

```
    2 FORMAT(6X,27HDATA ((RATES(I,J),I=1,6),J=I3,1H,I3,2H)/)
    4 FORMAT(4X,2H X1PE15.7,1H,1PE15.7,1H,1PE15.7,1H,1PE15.7,1H,)
    5 FORMAT(4X,2H X1PE15.7,1H,1PE15.7,1H/)
    6 FORMAT(6X,14HDATA (RATES(I,13,9H),I=1,6)/)
    7 FORMAT(4X,2H X1PE15.7,1H,1PE15.7,1H,1PE15.7,1H,1PE15.7,1H/)
      END
$IBFTC BRAKE.  LIST
      SUBROUTINE BRAKE(JA,JB,NRLIST,NR,NS,N2,NM)
      DIMENSION JA(12,1),JB(2),NRLIST(2)
      COMMON/DS/KK,K,NM
      COMMON/CT/KLA,KLB,IA,IB
      COMMON/SEE/JP(8,7)
      COMMON/DW/I,J,IT,MA,MB,MATEMP(4),MBTEMP(4),INR,JZ,LZ
      DATA ZP/6H/     /
      II=0
      DO 100 I=1,NR
      CALL PTORV(JA,I)
      IF(NRLIST(I).GE.7000)GO TO 30
      INR=I
      JW=1
      LS=1
      GO TO 80
   30 IF(II.NE.0)GO TO 35
      II=I-1
      JJ=0
   35 JM=JJ&3
      LS=JB(JM)
      JM=JB(JJ&2)
      JJ=JJ&JM
      JW=0
   40 JW=JW&1
      II=II&1
      INR=II
   80 DO 90 J=1,2
      JM=J-1
      JR=I&4*JM
      JS=I&5*JM
      JT=I&5*JM
      XX=JP(JR,I)
      JQ=JA(JT,I)
      IF(XX.LE.1.)GO TO 85
      WRITE(N2,2)INR,JS,JQ,INR,JS,ZP,XX
      GO TO 90
   85 WRITE(N2,2)INR,JS,JQ,INR,JS
   90 CONTINUE
      IF(JW.LT.LS)GO TO 40
  100 CONTINUE
      IF(NM.EQ.0)GO TO 105
      WRITE(N2,106)
  105 DO 200 J=1,NS
      KLA=0
      KLB=0
      NM=0
      II=0
      IA=0
```

```
      IB=0
      DO 150 I=1,NR
      IBETA=NRLIST(I)/7000
      IF(IBETA.EQ.1)GO TO 110
      INR=I
      JW=1
      GO TO 120
  110 IF(II.NE.0)GO TO 115
      II=I-1
      JJ=0
  115 JM=JJ&3
      JW=JB(JM)
      JM=JB(JJ&2)
      JJ=JJ&JM
      INR=II
  120 DO 130 JQ=1,JW
      INR=INR&IBETA
      KK=1
      CALL STAB(JA,N2,1)
      IF(IT.EQ.1)GO TO 135
      KK=6
      CALL STAB(JA,N2,2)
  130 CONTINUE
  135 IF(II.EQ.0)GO TO 150
      II=II&JW
  150 CONTINUE
      NM=1
      IF(IA.EQ.0)GO TO 151
      CALL STAB(JA,N2,3)
  151 IF(IB.EQ.0)GO TO 160
      CALL STAB(JA,N2,4)
  160 IF(KLA.EQ.0.AND.KLB.EQ.0)GO TO 185
      IF(KLA.EQ.0)GO TO 175
      IF(KLB.EQ.0)GO TO 165
      WRITE(N2,101)J
      GO TO 200
  165 WRITE(N2,201)J
      GO TO 200
  175 WRITE(N2,301)J
      GO TO 200
  185 WRITE(N2,401)J
  200 CONTINUE
      RETURN
    2 FORMAT(6X,3HPL(I3,1H,I3,4H)=Y(I3,5H)*PL(I3,1H,I3,1H),A1,F2.3)
  101 FORMAT(6X,2HB(I3,6H) = AA - BB)
  106 FORMAT(3X,11H30 CONTINUE)
  201 FORMAT(6X,2HB(I3,6H) = AA)
  301 FORMAT(6X,2HB(I3,6H) = - BB)
  401 FORMAT(6X,2HB(I3,6H) = 0.)
      END
$IBFTC STAB.   FULIST
      SUBROUTINE STAB(JA,N2,IS)
      DIMENSION JA(12,1)
      COMMON/DW/I,J,IT,MA,MB,MATEMP(4),MBTEMP(4),INR,JZ,LZ
      COMMON/DS/KK,K,NM
      COMMON/DT/KLA,KLB,IA,IB
      DIMENSION LAJ(4),LAK(4),LAP(4),LBJ(4),LBK(4),LBP(4)
      IF(NM.EQ.1) GO TO 5
      IT=0
      CALL ANUM(JA,2)
    5 GO TO (10,20,*C8,499),I5
   10 JZ=JZ-LZ
      IF(JZ)15,990,499
   15 JZ=-JZ
      GO TO 496
   20 JZ=JZ-LZ
      IF(JZ)25,990,495
   25 JZ=-JZ
      GO TO 499
  990 IT=1
 1000 RETURN
  498 CALL ABPRINE(N2,KLA,IA,LAJ,LAK,LAP,0)
      GO TO 1000
  499 CALL ABPRINE(N2,KLB,IB,LBJ,LBK,LBP,2)
      GO TO 1000
      END
$IBFTC ABPR.    FULIST
      SUBROUTINE ABPRINE(N2,KLAB,IAB,LABJ,LABK,LABP,NK)
C        KLAB = COUNT OF PL(M,N) IN NM
      DIMENSION LABJ(4),LABK(4),LABP(4)
      COMMON/DW/I,J,IT,MA,MB,MATEMP(4),MBTEMP(4),NR,JZ,LZ
      COMMON/DS/KK,K,NN
      DIMENSION PP(4),RP(4),PQ(6),TP(4)
      DATA SM1/6H)      /,SM2/6H)E     /,(PQ(JQ),JQ=1,6)/6H        ,6H2.*
     1,6H3.*    ,6H4.*    ,6H5.*    ,6H6.*   /
      DATA(TP(JQ),JQ=1,4)/6HA=    ,6HAA=AAE,6HBB=    ,6HBB=BBE/
C        NN = 0  DATA TO BE STORED
C        NN = 1  NO DATA TO BE STORED
      IF(NN.EQ.1.OR.IAB.EQ.4)GO TO 20
   10 IAB=IAB&1
      LABK(IAB)=KK
      LABJ(IAB)=IKK
      LABP(IAB)=JZ
   15 RETURN
   20 KB=0
   23 KB=KB&1
      IS=LABP(KB)
      PP(KB)=PQ(IS)
      IF(KB.EQ.IAB)GO TO 24
      RP(KB)=SM2
      GO TO 23
   24 RP(KB)=SM1
      IW=IGNK
      IZ=2GNK
      IF(KLAB.EQ.0)GO TO 40
      WRITE(N2,4)TP(IZ),(PP(JQ),LABJ(JQ),LABK(JQ),RP(JQ),JQ=1,IAB)
   25 IAB=0
      IF(NN)15,10,15
   40 WRITE(N2,5)TP(IW),(PP(JQ),LABJ(JQ),LABK(JQ),RP(JQ),JQ=1,IAB)
      KLAB=1
      GO TO 25
    4 FORMAT(6X,A6,4(A3,3HPL(I3,1H,I3,A2))
    5 FORMAT(6X,A3,4(A3,3HPL(I3,1H,I3,A2))
      END
$IBFTC ANUMJ.   FULIST
```

```
      SUBROUTINE ANUMJA(JA,NN)
      DIMENSION JA(12,1)
      COMMON/DW/I,J,IT,MA,MB,MATEMP(4),MBTEMP(4),NR,JZ,LZ
C
C      NN=1 NUMBER OF DIFFERENT SPECIES ON EACH SIDE OF EQUATIONS ARE
C                  COUNTED AND STORED
C      NN=2 COUNTS NUMBER OF TIMES SPECIES J APPEARS ON EACH SIDE OF
C                  EQUATION
C
      GO TO(100,200),NN
  100 MA=0
      MB=0
      DO 5 IQ=1,4
      MATEMP(IQ)=0
    5 MBTEMP(IQ)=0
      MA=1
      MATEMP(1)=JA(1,I)
      DO 10 K=2,6
      IF(JA(K,I).EQ.MATEMP(MA))GO TO 10
      IF(JA(K,I).EQ.0)GO TO 11
      MA=MA&1
      MATEMP(MA)=JA(K,I)
   10 CONTINUE
   11 MB=1
      MBTEMP(1)=JA(7,I)
      DO 20 K=8,12
      IF(JA(K,I).EQ.MBTEMP(MB))GO TO 20
      IF(JA(K,I).EQ.0)GO TO 21
      MB=MB&1
      MBTEMP(MB)=JA(K,I)
   20 CONTINUE
   21 RETURN
  200 JZ=0
      LZ=0
      DO 30 JJ=1,6
      IF(JA(JJ,I).NE.J)GO TO 30
      JZ=JZ&1
   30 CONTINUE
      DO 70 JJ=7,12
      IF(JA(JJ,I).NE.J)GO TO 70
      LZ=LZ&1
   70 CONTINUE
      GO TO 21
      END
$IBFTC DERV.    FULIST
      SUBROUTINE DERV(JA,JB,NRLIST,NR,N2)
      DIMENSION JA(12,1),JB(2),NRLIST(2)
      COMMON/SEE/JP(8,7)
      COMMON/SEEK/LL,KP,LP,KS,JS,KR,MP
      COMMON/VVVV/IRRAY(6),INR,IJ,J,XX,L,M,MJ,I
      COMMON/SAM/KK
      IS=0
      DO 250 I=1,NR
      CALL PTDRV(JA,I)
      IF(NRLIST(I).GE.7000)GO TO 70
      INR=1
      J=IGNRLIST(I)/5000
      KK=0
      CALL PRINT(JA,N2)
      GO TO 250
```

```
 70 IF(II.NE.0)GO TO 75
    II=I-1
    IJ=0
 75 IJ=3
    KP=0
    MS=0
    J=JJ&3
    NP=JB(J)
    LP=NP-1
    JJ=JJ&JB(JJ&2)
    DO 80 KC=1,LP
    LL=J&7
    KP=JB(LL)
    J=LL&KP
    CALL CPRINT(JB,N2)
 80 CONTINUE
    MS=1
    CALL CPRINT(JB,N2)
    DO 90 KK=1,NP
    II=III1
    INR=II
    CALL PRINT(JA,N2)
 90 CONTINUE
250 CONTINUE
    RETURN
    END
$IBFTC PRINT.  LIST
    SUBROUTINE PRIN7(JA,N2)
    DIMENSION JA(12,1)
    COMMON/SEE/JP(8,7)
    COMMON/VVVV/IRRAY(6),INR,IJ,JW,XK,L,N,NJ,I
    NJ=0
    DO 20 J=1,8
    XK=JP(J,1)
    IF(XK.NE.0.)GO TO 9
    IF(J.NE.1.AND.J.NE.5)GO TO 20
  9 L=JP(J,7)
    IF(L.EQ.0)GO TO 12
    DO 10 NN=2,6
    NNN=NN-1
 10 IRRAY(NNN)=JP(J,NN)
 12 NR=J-6
    JW=JLN-1
    CALL PHCRD(N2)
 20 CONTINUE
    IF(IJ.EQ.1)GO TO 50
    NJ=1
    XK=1.
    DO 40 N=1,2
    JW=5*N
    NN=4*N-3
    L=JP(NN,7)&1
    LL=1L6*(N-1)
    LLL=LLCS
    NJ=0
```

```
    DO 25 NN=LL,LLL
    NJ=NOL1
 25 IRRAY(NJ)=JA(NN,I)
    CALL PHCRD(N2)
 40 CONTINUE
 50 RETURN
    END
$IBFTC PHCRD.  FULIST
    SUBROUTINE PHCRD(N2)
    COMMON/VVVV/IRRAY(6),INR,IJ,J,XK,L,N,NJ,I
    COMMON/SAM/XX
    DIMENSION WP(7)
    DATA SP/6H)     /
   1,S2/6H)*X    /
   2,(WP(NN),NN=1,7)/4*6H)*YI   ,6H *YI   ,2*6H)*YI   /
   3,ZP/6H)*C(   /
    IF(L.EQ.0)GO TO 200
    NP=L((IJ/2)*1)-NJ)
    GO TO (25,90,100),IJ
 25 NP=SP
 30 IF(L.GE.5) GO TO 60
    IF(XK.EQ.1.) GO TO 40
    WRITE(N2,3)INR,J,XK,INR,N,NP,(WP(NN),IRRAY(NN),NN=1,L),NP
500 RETURN
 40 WRITE(N2,5)INR,J,INR,N,NP,(WP(NN),IRRAY(NN),NN=1,L),NP
    GO TO 500
 60 IF(XK.EQ.1.) GO TO 70
    WRITE(N2,3)INR,J,XK,INR,N,NP,(WP(NN),IRRAY(NN),NN=1,4),SP
 65 WRITE(N2,6)(WP(NN),IRRAY(NN),NN=5,L),NP
    GO TO 500
 70 WRITE(N2,5)INR,J,INR,N,NP,(WP(NN),IRRAY(NN),NN=1,4),SP
    GO TO 65
 90 IF(NJ.EQ.1)GO TO 25
    NP=S2
    GO TO 30
100 IF(NJ.EQ.1)GO TO 25
    IF(L.GE.4) GO TO 150
    IF(XK.EQ.1.) GO TO 130
    WRITE(N2,3)INR,J,XK,INR,N,NP,(WP(NN),IRRAY(NN),NN=1,L),ZP,XX,SP
    GO TO 500
130 WRITE(N2,5)INR,J,INR,N,NP,(WP(NN),IRRAY(NN),NN=1,L),ZP,XX,SP
    GO TO 500
150 IF(XK.EQ.1.) GO TO 170
    WRITE(N2,3)INR,J,XK,INR,N,NP,(WP(NN),IRRAY(NN),NN=1,3),SP
160 WRITE(N2,6)(WP(NN&1),IRRAY(NN),NN=4,L),ZP,XX,SP
    GO TO 500
170 WRITE(N2,5)INR,J,INR,N,NP,(WP(NN),IRRAY(NN),NN=1,3),SP
    GO TO 160
200 GO TO (210,230,250),IJ
210 WRITE(N2,4)INR,J,INR,N,SP
    GO TO 500
230 WRITE(N2,7)INR,J,INR,N,S2
    GO TO 500
250 WRITE(N2,7)INR,J,INR,N,ZP,XX,SP
    GO TO 500
  3 FORMAT(6X,3HPL(I3,1H,I3,3H) =F3.0,4H*QK(I3,1H,I3,6H)*RND(I1,5(A6,I
   13))
  4 FORMAT(6X,3HPL(I3,1H,I3,7H) = QK(I3,1H,I3,A3)
  5 FORMAT(6X,3HPL(I3,1H,I3,7H) = QK(I3,1H,I3,6H)*RND(I1,5(A6,I3))
  6 FORMAT(6H     X,4(A6,I3))
```

```
    7 FORMAT(6X,3HPL(I3,1H,I3,7H) = QK(I3,1H,I3,7H)PRMB(1,A4,I5,A3)
      END
$IBFTC CPRIN.  FULIST
      SUBROUTINE CPRINT(JB,N2)
      COMMON/SEEK/LL,KP,LP,MS,JS,KR,NP
      COMMON/SP/KK
      DIMENSION JC(2),VP(6),VM(6),RP(7),RM(7),IB(7)
      DATA SP/6H)      /
     1,(VP(MM),MM=1,6)/6H) = Y(,5*6H) & Y(/
     2,(VM(MM),MM=1,6)/6HX - C(,5*6H) - C(/
     3,(RP(MM),MM=1,7)/6H  & Y(,6*6H) & Y(/
     4,(RM(MM),MM=1,7)/6H  - C(,6*6H) - C(/
C
C        MS = 0  DESIGNATED CATALYST
C        MS = 1  ALL OTHER CATALYSTS
C
      ML=0
      IF(MS.NE.0) GO TO 50
      MM=LL
      KN=KP
      KPP=KN/7
      IF(KPP.EQ.0) GO TO 110
      WRITE(N2,5)KK,VP(1),JB(MM&1),VP(2),JB(MM&2),VP(3),JB(MM&3),VP(4),J
     1B(MM&4),VP(5),JB(MM&5),VP(6),JB(MM&6),SP
      MM=MM-1
      KN=KN&1
   20 MM=MM&7
      KN=KN-7
   22 ML=ML&1
      KPP=KN/8
      IF(KPP.EQ.0) GO TO 130
      IF(ML.GE.20) GO TO 200
      WRITE(N2,6)RP(1),JB(MM&1),RP(2),JB(MM&2),RP(3),JB(MM&3),RP(4),JB(M
     1M&4),RP(5),JB(MM&5),RP(6),JB(MM&6),RP(7),JB(MM&7),SP
      GO TO 20
   50 KN=LP
      KPP=KN/7
      IF(KPP.EQ.0) GO TO 160
      WRITE(N2,7)NP,(VM(JQ),JQ,JQ=1,6),SP
      MM=-1
      KN=KN&1
   70 MM=MM&7
      KN=KN-7
   72 ML=ML&1
      KPP=KN/8
      IF(KPP.EQ.0) GO TO 180
      DO 80 JQ=1,7
      LS=MM&JQ
   80 IB(JQ)=LS
      IF(ML.GE.20) GO TO 300
      WRITE(N2,8)(RM(JQ),IB(JQ),JQ=1,7),SP
      GO TO 70
  100 RETURN
  110 IF(KN.EQ.0)GO TO 100
      DO 115 JQ=1,KN
```

```
      LS=MM&JQ
  115 IA(JQ)=JB(LS)
      WRITE(N2,5)KK,(VP(JQ),IB(JQ),JQ=1,KN),SP
      GO TO 100
  130 IF(KN.EQ.0)GO TO 100
      DO 135 JQ=1,KN
      LS=MM&JQ
  135 IB(JQ)=JB(LS)
      IF(ML.GE.20) GO TO 250
      WRITE(N2,6)(RP(JQ),IB(JQ),JQ=1,KN),SP
      GO TO 100
  160 IF(KN.EQ.0) GO TO 100
      WRITE(N2,7)NP,(VM(JQ),JQ,JQ=1,KN),SP
      GO TO 100
  180 IF(KN.EQ.0) GO TO 100
      DO 185 JQ=1,KN
      LS=MM&JQ
  185 IB(JQ)=LS
      IF(ML.GE.20) GO TO 350
      WRITE(N2,6)(RM(JQ),IB(JQ),JQ=1,KN),SP
      GO TO 100
  200 ML=0
      WRITE(N2,8)KK,KK,RP(2),JB(MM&1),RP(3),JB(MM&2),RP(4),JB(MM&3),RP(5
     1),JB(MM&4),RP(6),JB(MM&5),SP
  210 MM=MM&5
      KN=KN-5
      GO TO 22
  300 ML=0
      WRITE(N2,9)(RM(JQ),IB(JQ),JQ=1,5),SP
  310 MM=MM&5
      KN=KN-5
      GO TO 72
  250 IF(KN.GE.6) GO TO 275
      WRITE(N2,8)KK,KK,(RP(JQ),IB(JQ),JQ=1,KN),SP
      GO TO 100
  350 IF(KN.GE.7)GO TO 375
      WRITE(N2,9)(RM(JQ),IB(JQ),JQ=1,KN),SP
      GO TO 100
  275 ML=0
      WRITE(N2,8)KK,KK,RP(2),IB(1),(RP(JQ),IB(JQ),JQ=2,5),SP
      GO TO 210
  375 ML=0
      WRITE(N2,9)(RM(JQ),IB(JQ),JQ=1,5),SP
      GO TO 310
    5 FORMAT(6X,2HC(I3,7(A6,I3))
    6 FORMAT(6H     X,8(A6,I3))
    7 FORMAT(6X,2HC(I3,4H) = ,7(A6,I3))
    8 FORMAT(6X,2HC(I3,6H) = C(I3,6(A6,I3))
    9 FORMAT(6X,5HX = X,7(A6,I3))
      END
$IBFTC PTGRV.  FULIST
      SUBROUTINE PTGRV(JA,II)
      DIMENSION JA(12,1)
      COMMON/SEE/JP(8,7)
      I=II
      DO 5 L=1,8
      DO 5 J=1,7
    5 JP(L,J)=0
      N=6
      L=0
      J=0
    8 J=J&1
      K=0
   10 K=K&1
      L=L&1
      IF(L.EQ.M)GO TO 15
      IF(JA(L,1).EQ.JA(L&1,1))GO TO 10
   15 KK=1
      JK=M-5
      JP(J,KK)=K
   16 IF(JK-L)20,30,25
   20 JP(J,KK&1)=JA(JK,1)
      GO TO 30
   25 JP(J,KK)=JA(JK,1)
   30 IF(KK.EQ.6)GO TO 35
      KK=KK&1
      JK=JK&1
      GO TO 16
   35 IF(L.EQ.M)GO TO 40
      IF(JA(L&1,1))8,40,8
   40 IF(M.EQ.12)GO TO 50
      M=12
      L=6
      J=4
      GO TO 8
   50 DO 80 J=1,8
      IF(JP(J,1).EQ.0)GO TO 80
      IF(JP(J,2).NE.0)GO TO 55
      JP(J,1)=6
      GO TO 80
   55 DO 60 K=2,6
      IF(JP(J,K))60,70,60
   60 CONTINUE
      JP(J,7)=5
      GO TO 80
   70 JP(J,7)=K-2
   80 CONTINUE
      RETURN
      END
```

APPENDIX "B"

LISTING OF THE INTEGRATION PROGRAM

```
$JOB            GENERAL APPLIED SCIENCE            SCHECHTER
$IBFTC PSTPA    FULIST
      DIMENSION F(31),CI(31),C(31),R(31),D(31),SF(31),AM(31,31),AC(31,31
     1),P(3)
      COMMON/CHPG/AM,CI,F,U,RHO,TP,IT
   10 READ(5,500)TN,U,SS,EL,SMAX,DTM,SMX,S,SMIN,SPRNT,DT,M
      WRITE(6,800)
      CALL SPOUT(M)
      READ(5,500)(C(I),I=1,M)
      N=0
      SH=0.5*S
      DTM=0.5*DTM
      CALL FW(C,CI,0)
      IF(N)30,20,30
   20 CALL FT(C,TN,H,0)
   30 HH=H&0.5*U*U
   40 CALL FT(CI,TN,H,1)
      IF(N)45,80,45
   45 IF(L)50,60,50
   50 TT=(TN-TP)-DT
      IF(ABS(TT)/TN-.0001)60,210,210
   60 DT=TN-TP
      IF(ABS(DT)/TP.GT.DTM)GO TO 280
      IF(ABS(DT)/TP.GT.DTM)GO TO 70
      SD=2.0*S
      IF(SD.GT.SMX)GO TO 70
      SH=S
      S=SD
   70 IF(SS-PSMX)90,80,80
   80 READ(5,500)P(1),P(2),P(3),PSMX
      TPR=P(3)&P(3)
   90 PP=P(1)&SS*(P(2)&SS*P(3))
      L=0
      CALL FX(CI,TN,PP,RHO,1)
      IF(N)130,150,140
  140 IF(SS-PRNT)160,150,150
  150 PRNT=SPRNT&SS
      X=SS*EL
      CALL FW(CI,C,1)
      WRITE(6,600)SS,X,H,TN,U,PP,RHO,S,N,(I,C(I),I=1,M)
      IF(SS.GE.SMAX)GO TO 10
  160 N=N&1
      CALL FG
      DO 170 I=1,M
      SF(I)=F(I)
      IF(IT.NE.0)F(I)=SF(I)&DT*D(I)
  170 R(I)=CI(I)
      TP=TN
      SU=U
      SSS=SS
  180 F1=-TPR/(EL*RHO)
      S1=-P(2)/(EL*RHO)
      U1=1.0-SH*SH*F1/EL
      G1=SS&SH*F1*SSS*S1
      G2=SH*U/EL&SS
      U=(G1&SH*F1*G2)/U1
      SS=(G2&SH*G1/EL)/U1
      H=HH-0.5*U*U
  190 CALL PADE(AM,F,AC,S,M,31)
```

```
      DO 200 I=1,M
      CI(I)=R(I)&F(I)
  200 IF(CI(I).LT.0.0)GO TO 280
      GO TO 40
  210 L=L&1
      IF(L-10)220,220,240
  220 IF(L-1)250,230,250
  230 DP=DT
      FP=TT
      DT=TN-TP
      GO TO 260
  240 WRITE(6,900)TN,DT,TP
      STOP
  250 Q=(DT-DP)*TT/(TT-FP)
      DP=DT
      FP=TT
      DT=DT-Q
  260 DO 270 I=1,M
  270 F(I)=SF(I)&DT*D(I)
      GO TO 190
  280 S=SH
      IF(S-SMIN)310,290,290
  290 SH=0.5*S
      DT=0.5*DT
      TN=TP
      U=SU
      SS=SSS
      DO 300 I=1,M
      F(I)=SF(I)
      IF(IT.NE.0)F(I)=SF(I)&DT*D(I)
  300 CI(I)=R(I)
      L=0
      GO TO 180
  310 WRITE(6,1000)
      STOP
  500 FORMAT(7E10.6)
  600 FORMAT(1H0/3HPS=E16.8,10X2HX=E16.8,10X2HH=E16.8,10X2HT=E16.8/3H U=
     1E16.8,10X2HP=E16.8,10X4HRHO=E16.8,10X5HSTEP=E15.8,4X2HN=I3//(6(I5,
     2E14.7)))
  800 FORMAT(1H1//1H040X24HPADE STREAM TUBE PROGRAM//)
  900 FORMAT(30H0TEMPERATURE DOES NOT CONVERGE/3H T=E15.8,10X2HDT=E15.8,
     110X2HP=E15.8)
 1000 FORMAT(20H0STEP SIZE LESS THAN MINIMUM)
      END
```

```
$IBFTC PT311   FULIST
      SUBROUTINE PT(A,TEMP,SIGM,KOPT)
      DIMENSION A(31)
C     PROGRAM 2    T3A    7/20/66
      DIMENSION SPECT7X(  11),WT7X(  11),SYM7X(  11,2)
      INTEGER SPECT7X
      DATA         (SPECT7X(I),WT7X(I),SYM7X(I,1),SYM7X(I,2),I=  1, 11)/
     X   1,3.20000E 01,6H     O,6H2     ,
     X   2,1.60000E 01,6H     O,6H      ,
     X   3,2.80160E 01,6H     N,6H2     ,
     X   4,1.40080E 01,6H     N,6H      ,
     X   5,3.00080E 01,6H     N,6HO     ,
     X   6,3.00080E 01,6H    NO,6H+     ,
     X   7,5.48620E-04,6H     E,6H-     ,
     X  58,1.60000E 01,6H     O,6H+     ,
     X  60,3.20000E 01,6H    O2,6H+     ,
     X  57,1.40080E 01,6H     N,6H+     ,
     X  59,2.80160E 01,6H    N2,6H+     /
      DATA NORD7X,NFIT7X/    3,    6/
      DIMENSION ENT7X(  11,    7),HCOE7X(  11,    3,    6)
      DATA (ENT7X(I,    1),I=    1,  11)/
     X 1.00000E 02, 1.00000E 02, 1.00000E 02, 1.00000E 02, 1.00000E 02,
     X 2.00000E 02, 1.00000E 02, 1.00000E 02, 2.00000E 02, 2.00000E 02,
     X 1.00000E 02/
      DATA (ENT7X(I,    2),I=    1,  11)/
     X 1.25000E 03, 5.00000E 02, 1.80000E 03, 3.30000E 03, 6.00000E 02,
     X 2.00000E 03, 3.00000E 04, 4.00000E 03, 1.40000E 03, 4.00000E 02,
     X 4.00000E 02/
      DATA (ENT7X(I,    3),I=    1,  11)/
     X 4.50000E 03, 2.50000E 03, 3.70000E 03, 8.50000E 03, 1.30000E 03,
     X 7.00000E 03, 3.00000E 04, 6.00000E 03, 2.50000E 03, 2.50000E 03,
     X 2.00000E 03/
      DATA (ENT7X(I,    4),I=    1,  11)/
     X 8.80000E 03, 1.47500E 04, 8.50000E 03, 1.60000E 04, 3.00000E 03,
     X 9.50000E 03, 3.00000E 04, 1.00000E 04, 5.00000E 03, 4.00000E 03,
     X 3.50000E 03/
      DATA (ENT7X(I,    5),I=    1,  11)/
     X 1.22500E 04, 1.77500E 04, 1.05000E 04, 2.45000E 04, 1.52500E 04,
     X 1.20000E 04, 3.00000E 04, 1.30000E 04, 6.00000E 03, 7.50000E 03,
     X 4.50000E 03/
      DATA (ENT7X(I,    6),I=    1,  11)/
     X 1.60000E 04, 2.40000E 04, 1.60000E 04, 2.70000E 04, 2.00000E 04,
     X 1.50000E 04, 3.00000E 04, 1.45000E 04, 7.50000E 03, 2.00000E 04,
     X 5.50000E 03/
      DATA (ENT7X(I,    7),I=    1,  11)/
     X 3.00000E 04, 3.00000E 04, 1.80000E 04, 3.00000E 04, 3.00000E 04,
     X 1.75000E 04, 3.00000E 04, 3.00000E 04, 1.00000E 04, 3.00000E 04,
     X 1.00000E 04/
      DATA (HCOE7X(I,    1,    1),I=    1,  11)/
     X 3.03460E 04, 1.65890E 08, 3.65660E 04, 3.61740E 08, 3.22550E 07,
     X 3.54020E 08, 0.          , 1.04920E 09, 3.51230E 08, 1.43950E 09,
     X 5.77040E 08,
      DATA (HCOE7X(I,    2,    1),I=    1,  11)/
     X 5.37170E 03, 1.61880E 04, 1.07300E 04, 1.59220E 04, 1.01730E 04,
     X 1.06150E 04, 4.07580E 08, 1.39700E 04, 9.28900E 03, 1.74290E 04,
     X 1.11420E 04/
      DATA (HCOE7X(I,    3,    1),I=    1,  11)/
     X 1.15420E 00,-2.00760E 00, 7.60540E-01, 2.55240E-02, 6.95200E-01,
     X 7.07450E-01, 0.          , 6.09000E-03, 1.02070E 00,-1.61760E 00,
```

```
     X 1.31050E-01/
      DATA (HCOE7X(I,    1,    2),I=    1,  11)/
     X-1.19010E 06, 1.66400E 08,-2.32110E 06, 3.68540E 08, 3.24870E 07,
     X 3.51210E 08, 0.          , 1.05290E 09, 3.89540E 08, 1.43970E 09,
     X 5.77790E 08/
      DATA (HCOE7X(I,    2,    2),I=    1,  11)/
     X 1.13420E 04, 1.41540E 04, 1.33490E 04, 1.18020E 04, 9.39500E 03,
     X 1.28240E 04, 4.07580E 08, 1.24090E 04, 1.15960E 04, 1.61740E 04,
     X 1.03970E 04/
      DATA (HCOE7X(I,    3,    2),I=    1,  11)/
     X 3.33060E-01,-3.36210E-02, 1.38800E-01, 6.49700E-01, 1.34050E 00,
     X 8.52290E-02, 0.          , 2.01260E-01, 1.60480E-01,-4.90010E-02,
     X 1.06230E 00/
      DATA (HCOE7X(I,    1,    3),I=    1,  11)/
     X-3.86140E 06, 1.67320E 08,-3.26450E 04, 3.33070E 08, 3.05040E 07,
     X 3.71950E 08, 0.          , 1.06340E 04, 3.89230E 08, 1.44110E 09,
     X 5.76420E 08,
      DATA (HCOE7X(I,    2,    3),I=    1,  11)/
     X 1.25750E 04, 1.34180E 04, 1.35700E 04, 2.01500E 04, 1.24500E 04,
     X 6.89810E 03, 4.07580E 08, 8.92670E 03, 1.19410E 04, 1.50930E 04,
     X 1.17690E 04/
      DATA (HCOE7X(I,    3,    3),I=    1,  11)/
     X 1.92940E-01, 1.13550E-01, 6.85070E-02, 1.58740E-01, 1.64920E-01,
     X 5.08490E-01, 0.          , 4.91480E-01, 1.11450E-01, 1.57090E-01,
     X 7.19190E-01/
      DATA (HCOE7X(I,    1,    4),I=    1,  11)/
     X-1.92270E 07, 2.55590E 08, 2.52480E 07, 6.15770E 08, 3.02900E 07,
     X 3.95040E 08, 0.          , 1.04290E 09, 3.91780E 08, 1.44130E 09,
     X 5.72860E 08/
      DATA (HCOE7X(I,    2,    4),I=    1,  11)/
     X 1.61080E 04, 1.44870E 03, 6.21570E 03,-1.51870E 04, 1.25870E 04,
     X 1.19650E 03, 4.07500E 08, 1.30180E 04, 1.09220E 04, 1.49790E 04,
     X 1.38030E 04/
      DATA (HCOE7X(I,    3,    4),I=    1,  11)/
     X-1.70880E-02, 5.19300E-01, 5.18750E-01, 1.26300E 00, 1.44090E-01,
     X 8.08560E-01, 0.          , 2.06870E-01, 2.13390E-01, 1.01390E-01,
     X 4.28640E-01/
      DATA (HCOE7X(I,    1,    5),I=    1,  11)/
     X-2.84900E 07, 4.35670E 08, 4.80820E 07, 1.57260E 08, 2.56200E 06,
     X 3.57880E 08, 0.          , 9.91050E 08, 3.98160E 08, 1.43360E 09,
     X 5.63040E 08/
      DATA (HCOE7X(I,    2,    5),I=    1,  11)/
     X 1.76530E 04,-1.88410E 04, 2.62830E 03, 2.22410E 04, 1.62250E 04,
     X 2.22210E 03, 4.07590E 08, 2.08840E 04, 8.79760E 03, 1.70380E 04,
     X 1.81590E 04/
      DATA (HCOE7X(I,    3,    5),I=    1,  11)/
     X-7.44410E-02, 1.09080E 00, 6.89620E-01, 4.99190E-01, 2.40310E-02,
     X 7.65840E-01, 0.          ,-1.56500E-02, 3.90460E-01, 4.41360E-02,
     X-5.53280E-02/
      DATA (HCOE7X(I,    1,    6),I=    1,  11)/
     X-4.88450E 07, 6.72300E 07,-3.26560E 07,-2.93760E 08,-6.40290E 07,
     X 2.38320E 08, 0.          , 9.81050E 08, 4.07670E 08, 1.46870E 09,
     X 5.62040E 08/
      DATA (HCOE7X(I,    2,    6),I=    1,  11)/
     X 2.02380E 04, 7.96170E 03, 1.27200E 04, 5.56510E 04, 2.25340E 04,
     X 2.28510E 04, 4.07500E 08, 2.23740E 04, 6.25950E 03, 1.38250E 04,
     X 1.05290E 04/
      DATA (HCOE7X(I,    3,    6),I=    1,  11)/
     X-1.56510E-01, 5.75410E-01, 3.74230E-01,-1.19300E-01,-1.43040E-01,
     X 7.88830E-02, C.          ,-6.78150E-C2, 5.5966CE-01, 1.24450E-01,
     X-8.89430E-C2/
      DIMENSICN A7X(  11(,B7X(  11)
      DATA NSPC7X/  1(/
      IF(NOPT .NE. 5) GC TC 9C0
      T7X=TEMP
C     ENTHALPY NANIEC
      BIGH=C.
C     LOCP FCR EACH SPECIES
      DC 27C I7X=1,NSPC7X
C     FIND APPROPRIATE FIT FCR H=F(T)
      DC 255 K7X=1,NFIT7X
      IF(T7X.GE.ENT7X(I7X,K7X).AND.T7X.LE.ENT7X(I7X,K7X+1)
     1) GC TC 256
  255 CCNTINUE
      CALL ERRCR(T7X,I7X,1(
      K7X=NFIT7X
  256 CCNTINUE
      H77X=C.
      CC 258 M7X=1,NCRD7X
      MP7X=NCRD7X+1-M7X
      H77X=H77X*T7X+HCOE(X(I7X,MP7X,K7X)
  258 CCNTINUE
      BIGH=BIGH+H77X*A((7X)
  27C CCNTINUE
      RETURN
      ENTRY FCP(2,TEMP,BIGC,SMALLC(
      DIMENSICN SMALLC(31)
      T7X=TEMP
C     CP NANIEC
      BIGC=C.
      DC 29C I7X=1,NSPC7X
C     FIND APPRCPRIATE FIT FCR CP=F(T)
      CC 275 K7X=1,NFIT7X
      IF(I7X.GE.ENT7X(I7X,K7X(.AND.I7X.LE.ENT7X(I7X,K7X+1(
     1)GC TC 276
  275 CCNTINUE
      CALL ERRCR(T7X,I7X,3)
      K7X=NFIT7X
  276 CCNTINUE
      H77X=C.
      IF(NCRDC7X.EC.1)GO TO 280
      CC 278 M7X=2,NCRD7X
      MM7X=NCRD7X+2-M7X
      H77X=H77X*T7X+HCOE7X(I7X,MM7X,K7X)*FLCAT(MM7X-1(
  278 CCNTINUE
  28C CCNTINUE
      SMALLC(I7X)=H77X
      BIGC=BIGC+H77X*A((7X(
  290 CCNTINUE
      RETURN
      ENTRY FH(C,C,MCPT)
      DIMENSICN C(31),C(31)
      IF(MCPT)3,C,3CC,32G
  3CC CCNTINUE
      DC 31C I7X=1,NSPC7X
      C(I7X)=C(I7X)/W77X(I7X(
  31C CCNTINUE
      RETURN
```

```
  32C CCNTINUE
      DC 33C I7X=1,NSPC7X
      C(I7X)=WT7X(I7X)*C(I7X)
  33C CCNTINUE
      RETURN
      ENTRY FX(B,T,P,RHC,NCPT(
      DIMENSICN B(31(
      F7X=C.
      IF(NCPT)42C,4CC,42C
  40C CCNTINUE
      DO 41C I7X=1,NSPC7X
      F7X=F7X+B(I7X(/WT7X(I7X(
  41C CCNTINUE
      GC TC 45C
  42C CCNTINUE
      DC 43C I7X=1,NSPC7X
      F7X=F7X+B(I7X)
  43C CCNTINUE
  45C CCNTINUE
      RHC=P/F7X/8.3146E4/T
      RETURN
  9C0 CCNTINUE
      DIMENSICN GX7X(3),XX7X(5)
      IF(TEMP.LE.C.)TEMP=273.
      DC 915 J7X=1,11
      XX7X(1)=TEMP
      GX7X(1)=-BIGH
      GX7X(2)=0.0
      GX7X(3)=0.C
      DC 904 I7X=1,NSPC7X
      C7X=A(I7X)
      IF(NOPT.GT.C) C7X=WT7X(I7X)*C7X
      DC 9C1 K7X=1,NFIT7X
      IF(XX7X(1).GE.ENT7X(I7X,K7X).AND.XX7X(1)
     1.LE.ENT7X(I7X,K7X+1))GC TO 9C2
  901 CCNTINUE
      CALL ERRCR(XX7X(1),I7X,4(
      K7X=NFIT7X
  9C2 DC 9C3 MM7X=1,3
  9C3 GX7X(MM7X)=GX7X(MM7X(+C7X*HCCE7X(I7X,MM7X,K7X)
  904 CCNTINUE
  90E XX7X(2)=0.5*GX7X(2)/GX7X(3)
      XX7X(3)=SQRT(XX7X(2)**2-GX7X(1)/GX7X(3))
      XX7X(4)=-XX7X(2)+XX7X(3)
      XX7X(5)=-XX7X(2(-XX7X(3)
      IF(XX7X(4).LE.C.0( GC TC 9C8
      IF(XX7X(5).LE.0.0( GC TC 907
      IF(ABS(XX7X(4(-XX7X(1((-ABS(XX7X(5)-XX7X(1()) 907,
     19C7,9C8
  9C7 TEMP=XX7X(4)
      GC TC 9C9
  9C8 TEMP=XX7X(5)
  909 IF(TEMP.LT.C.(CALL ERRCR(TEMP,J7X,7)
      IF(ABS((TEMP-XX7X(1))/XX7X(1()-1.E-6(920,915,915
  915 CCNTINUE
      CALL ERRCR(TEMP,11,6)
  92C CCNTINUE
      RETURN
      ENTRY SPCLT(NS7X(
      AS7X=ASPC7X
      WRITE(6,1000)(I7X,SYM7X(I7X,1),SYM7X(I7X,2),I7X=1,ASPC7X)
 1000 FCRMAT(1HC/(8(I3,2A6)))
      ENC


  241 CARCS PRINTED
```

58

```
$IBFTC ERR      FULIST
      SUBROUTINE ERROR(T,I,J)
      GC TC(50,1C,50,50,1C,20,30),J
   1C WRITE(6,1CO)
      STCP
   20 WRITE(6,2CO)
      STOP
   3C WRITE(6,3CC)
      STCP
   5C RETURN
  1CC FCRMAT(16HOUNCEFINED ERRCR)
  200 FORMAT(30HCTEMPERATURE DCES NCT CCNVERGE)
  30C FCRMAT(21HCNEGATIVE TEMPERATURE)
      ENC


    15 CARCS PRINTED
```

```
$IBFTC PG.    FULIST
      SUBROUTINE PG
C     PROGRAM 3    T3A    7/20/66
      COMMON/CMPG/A,Y,D,B,RHCB,TA,II
      DIMENSION Y(31),A(31,31),D(31),B(31),RATES(6,54)
      DIMENSION RHC(6),C(10),CK( 54,2),PL( 54,10)
      DATA ((RATES(I,J),I=1,6),J=  1, 12)/
     X  3.1G5C529E C1, -C.           ,  3.7750CCCE 04,  3.C3390716 01,
     X -0.           , -0.           ,  1.7575711E 01,  1.5000CCOE 00,
     X  1.51COOCCE C4,  1.9000467E 01,  1.5000CCOE 00,  3.0200CCCE 03,
     X  3.2?2933EE C1, -0.           ,  6.1600000E 04,  2.9933306E 01,
     X -C.           ,  4.CCC0CC0E 04,  1.8683C45E 01,  1.0000000E 00,
     X  3.15400CCE C4,  4.4442264E 01, -1.C000CC0E 00, -0.           ,
     X  2.4558325E C1,  5.CC0000C0E-01,  6.7800CC0E 04,  4.5439379E 01,
     X -1.CCC0CCCE CC,  0.           ,  1.9581425E 01,  1.0000CC0E 00,
     X  8.G659555E C4,  4.5033913E G1, -1.CC00CCCE 00,  0.           ,
     X  3.1729225E C1,  0.           ,  C.           ,  3.3233306E S1,
     X  C.           ,  2.3C00CCCE 04,  3.2645522E 01,  0.           ,
     X  0.           ,  3.3233306E C1,  0.           ,  3.5400C00E 04,
     X  3.1??5225E C1,  0.           ,  C.           ,  3.2134459E 01,
     X  0.           ,  4.C900CCCE 04,  3.3115523E G1,  C.           ,
     X  0.           ,  3.5130428E 01,  C.           ,  7.3099555E 04,
     X  3.3338686E C1,  0.           ,  C.           ,  2.9426640E 01,
     X  5.CCC00C0E-C1,  8.7159959E 04,  2.8222672E 01,  0.           ,
     X  C.           ,  2.7306384E C1,  C.           ,  1.2400000E 04/
      DATA ((RATES(I,J),I=1,6),J= 13, 24)/
     X  3.0119787E 01,  C.           ,  C.           ,  2.9026166E 01,
     X  C.           ,  1.7900?C0E 04,  3.0302110E 01,  0.           ,
     X  C.           ,  3.0812538E C1,  C.           ,  5.0100000E 04,
     X  2.9426646E 01,  0.           ,  3.00000CCE 03,  2.3799822E 01,
     X  5.CCC00CCE-C1,  1.4100CC0E 04,  3.2422372E 01,  C.           ,
     X  C.           ,  2.84C4992E 01,  5.CC00C00E-01,  5.8300CC0E 04,
     X  3.2675137E 01,  0.           ,  C.           ,  3.5340144E 01,
     X  0.           ,  3.2200CCCE 04,  4.1223308E C1,  0.           ,
     X  C.           ,  1.C00CCC0E 35,  C.           ,  0.           ,
     X  3.1543044E C1, -5.CC0CCC0E-01,  C.           ,  4.4442264E 01,
     X -2.0CCCC0CE CC,  1.3000C00E 04,  2.9933606E C1, -5.0000000E-01,
     X  3.5C000CC0E C3,  4.6234024E 01, -2.CC00CCCE 00,  5.5900CC0E 04,
     X  3.1543044E 01, -5.CCC0CCC0E-C1,  C.           ,  3.1415211E 01,
     X -5.CCCC0C0E-C1,  1.C2000C0E C4,  3.1543C44E 01, -5.0000000E-01,
     X  0.           ,  3.6C20381E C1, -1.C000000E 00,  2.8000000E 04,
     X  2.9533606E C1, -5.CC00CC0E-01,  C.           ,  2.4124463E 01,
     X  0.           ,  2.63C0000E C4,  3.1543C44E 01, -5.00G0000E-01,
     X  0.           ,  4.C06C237E 01, -1.C000C00E 00,  6.1C00000E 04/
      DATA ((RATES(I,J),I=1,6),J= 25, 36)/
     X  3.1032218E 01, -5.CCCCC0CCE-01,  C.           ,  1.9701615E C1,
     X  1.CCCCC0CCE CC,  2.4700C00E 04,  2.9533606E 01, -5.0000000E-01,
     X  C.           ,  3.1368051E C1, -5.CC00CC0E-01,  3.7700CC0E 04,
     X  7.1532218E 01, -5.CC00CC0E-C1,  0.           ,  4.1341171E C1,
     X -1.CC000CCE CC,  1.8C00C00E C3,  4.C019415E 01, -5.0000000E-01,
     X  7.55C0CCCE C4,  3.8633123E 01, -5.CC00C0CE-01, -0.           ,
     X  4.4582026E C1, -1.CCCCCCCE C0,  5.9400CCCE 04,  3.7483215E 01,
     X -5.CCCCCCC0E-C1, -0.           ,  4.5609183E 01, -1.00000CCE C0,
     X  5.94CC000E 04,  3.8801458E C1, -5.CCC0CCCE-01, -C.           ,
     X  4.2545144E 01, -1.CC0CCCCE 00,  5.94C0CCCE C4,  3.5455C67E 01,
     X -5.CCC00C0E-C1, -0.           ,  4.5C84118E 01, -1.00000C0E 00,
     X  1.1320CCCE C5,  4.2135679E C1, -1.C0000C0E 00, -C.           ,
     X  4.6314C65E C1, -1.CCC0CCCE C0,  1.1320000E 05,  4.3352441E C1,
     X -1.CCC00C0E CC, -0.           ,  4.439057CE 01, -1.000CCCCE 00,
```

```
     X  1.13200CCE C5,  4.2446531E 01, -1.CC00CC0E 0C, -0.           ,
     X  1.8830011E C1,  5.CCC0CC0E-01,  1.8100CCCE 05,  3.6782947E C1,
     X -1.CCC000CE CC,  0.           ,  1.7913714E 01,  5.0C0000CE-01,
     X  1.81000C0E C5,  3.5827436E 01, -1.CC00CC0E 0C,  C.           /
      DATA ((RATES(I,J),I=1,6),J= 37, 48)/
     X  1.7913714E C1,  5.CCC0CC0E-01,  1.8100CC0E 05,  3.5827436E 01,
     X -1.CCCC0C0E CC,  0.           ,  3.5106992E 01, -1.0000GCCE 00,
     X  1.8100CC0E C5,  5.7C81329E C1, -2.5C00CCCE 00,  C.           ,
     X  1.7008357E C1,  5.CCCC0C0E-C1,  1.4C00CCCE C5,  3.6163908E 01,
     X -1.CCC00C0E CC,  0.           ,  1.8176C84E C1,  5.0000000E-01,
     X  1.4CC00CCE C5,  3.652C583E C1, -1.00000C0E 00,  0.           ,
     X  1.7913714E C1,  5.C0CC000E-01,  1.4000CC0E 05,  3.6297439E 01,
     X -1.CCCCCC0E CC,  0.           ,  3.8732267E 01, -1.00C0CCCE G0,
     X  1.4CC00CCE C5,  5.7081329E 01, -2.5000CCCE 00,  C.           ,
     X  4.2826639E C1, -1.CC0000CE 00,  1.69000CCE C5,  6.0661575E 01,
     X -2.5CC00CCE CC,  0.           ,  4.4197667E 01, -1.0000000E 0C,
     X  1.65CCCC0E C5,  6.2C31335E 01, -2.5C00CCCE 00,  0.           ,
     X  4.6905717E C1, -1.CC000CCE C0,  1.6900000E 05,  6.473939CE 01,
     X -2.5CCC0CCE 00,  0.           ,  3.9224770E 01, -1.0C000CCE 00,
     X  1.65C0000E C5,  5.7081329E C1, -2.5CC0CCCE CC,  C.           ,
     X  4.12C1935E 01, -1.CCC0CCCE CC,  1.5800CCCE 05,  6.0661575E 01,
     X -2.5CCC0CCE CC,  0.           ,  4.2600843E 01, -1.00000C0E 00,
     X  1.58C0CC0E C5,  6.2031339E 01, -2.5C00CCCE 00,  0.           /
      DATA ((RATES(I,J),I=1,6),J= 49, 54)/
     X  4.5283374E C1, -1.CC0CC0CE 00,  1.5800CC0E 05,  6.473935CE 01,
     X -2.5CCC0CCE CC,  C.           ,  3.7615333E 01, -1.0000000E 00,
     X  1.58CCCC0E C5,  5.7C81329E 01, -2.5000CCCE 0C,  C.           ,
     X  4.0624858E C1, -1.CC00C0CE 0C,  1.C790CC0E 05,  6.0661575E C1,
     X -2.5CCC0CCE CC,  0.           ,  4.2004277E C1, -1.C000CCCE C0,
     X  1.C7900C0E C5,  6.2C31335E 01, -2.5C00CCCE 00,  C.           ,
     X  3.5255587E C1, -1.CCC0000E 00,  1.C790CC0E 05,  5.5277739E 01,
     X -2.5CCG0CCE CC,  C.           ,  3.7027542E 01, -1.00G0CCCE 00,
     X  1.C79CCC0E C5,  5.7C81329E 01, -2.5000CCCE 00,  0.           /
      I7=1
      RHC(1)=.515302*RHCB
      DC 5 I=2,6
    5 RHC(I)=RHC(I-1)*RHC(1)
      X=C.
      DC 6 I=1, 11
    6 X=X+Y(I)
      IF(ABS((TA-T)/TA)-.0CC1)20,2C,10
   1C T=TA
      TL=ALCG(T)
      DO 18 I=1, 54
      C=RATES(1,I)
      IF(C-1.E30)12,11,11
   11 CK(I,1)=C.
      GC TC 13
   12 C=C+RATES(2,I)*TL-RATES(3,I)/T
      CK(I,1)=EXP(C)
   13 C=RATES(4,I)
      IF(C-1.E30)15,14,14
   14 CK(I,2)=0.
      GC TC 18
   15 C=C+RATES(5,I)*TL-RATES(6,I)/T
      CK(I,2)=EXP(C)
   18 CCNTINUE
   2C DC 25 I=1, 11
      DC 25 J=1, 11
```

```
25 A(I,J)=0.
   INC=0
   PL(  1,  1) = GK(  1,  1)*RHC(1)*Y(  3)
   PL(  1,  2) = GK(  1,  1)*RHC(1)*Y(  2)
   PL(  1,  6) = GK(  1,  2)*RHC(1)*Y(  4)
   PL(  1,  7) = GK(  1,  2)*RHC(1)*Y(  5)
   PL(  2,  1) = GK(  2,  1)*RHC(1)*Y(  2)
   PL(  2,  2) = GK(  2,  1)*RHC(1)*Y(  5)
   PL(  2,  6) = GK(  2,  2)*RHC(1)*Y(  1)
   PL(  2,  7) = GK(  2,  2)*RHC(1)*Y(  4)
   PL(  3,  1) = GK(  3,  1)*RHC(1)*Y(  1)
   PL(  3,  2) = GK(  3,  1)*RHC(1)*Y(  3)
   PL(  3,  6) = 2.*GK(  3,  2)*RHC(1)*Y(  5)
   PL(  4,  1) = GK(  4,  1)*RHC(1)*Y(  2)
   PL(  4,  2) = GK(  4,  1)*RHC(1)*Y(  4)
   PL(  4,  6) = GK(  4,  2)*RHC(1)*Y(  7)
   PL(  4,  7) = GK(  4,  2)*RHC(1)*Y(  6)
   PL(  5,  1) = 2.*GK(  5,  1)*RHC(1)*Y(  4)
   PL(  5,  6) = GK(  5,  2)*RHC(1)*Y(  7)
   PL(  5,  7) = GK(  5,  2)*RHC(1)*Y( 11)
   PL(  6,  1) = 2.*GK(  6,  1)*RHC(1)*Y(  2)
   PL(  6,  6) = GK(  6,  2)*RHC(1)*Y(  7)
   PL(  6,  7) = GK(  6,  2)*RHC(1)*Y(  4)
   PL(  7,  1) = GK(  7,  1)*RHC(1)*Y(  2)
   PL(  7,  2) = GK(  7,  1)*RHC(1)*Y( 11)
   PL(  7,  6) = GK(  7,  2)*RHC(1)*Y(  8)
   PL(  7,  7) = GK(  7,  2)*RHC(1)*Y(  3)
   PL(  8,  1) = GK(  8,  1)*RHC(1)*Y(  2)
   PL(  8,  2) = GK(  8,  1)*RHC(1)*Y( 11)
   PL(  8,  6) = GK(  8,  2)*RHC(1)*Y(  4)
   PL(  8,  7) = GK(  8,  2)*RHC(1)*Y(  6)
   PL(  9,  1) = GK(  9,  1)*RHC(1)*Y(  1)
   PL(  9,  2) = GK(  9,  1)*RHC(1)*Y( 11)
   PL(  9,  6) = GK(  9,  2)*RHC(1)*Y(  9)
   PL(  9,  7) = GK(  9,  2)*RHC(1)*Y(  3)
   PL( 10,  1) = GK( 10,  1)*RHC(1)*Y(  5)
   PL( 10,  2) = GK( 10,  1)*RHC(1)*Y( 11)
   PL( 10,  6) = GK( 10,  2)*RHC(1)*Y(  4)
   PL( 10,  7) = GK( 10,  2)*RHC(1)*Y(  3)
   PL( 11,  1) = GK( 11,  1)*RHC(1)*Y(  1)
   PL( 11,  2) = GK( 11,  1)*RHC(1)*Y( 10)
   PL( 11,  6) = GK( 11,  2)*RHC(1)*Y(  2)
   PL( 11,  7) = GK( 11,  2)*RHC(1)*Y(  6)
   PL( 12,  1) = GK( 12,  1)*RHC(1)*Y(  3)
   PL( 12,  2) = GK( 12,  1)*RHC(1)*Y(  8)
   PL( 12,  6) = GK( 12,  2)*RHC(1)*Y(  4)
   PL( 12,  7) = GK( 12,  2)*RHC(1)*Y(  6)
   PL( 13,  1) = GK( 13,  1)*RHC(1)*Y(  1)
   PL( 13,  2) = GK( 13,  1)*RHC(1)*Y(  8)
   PL( 13,  6) = GK( 13,  2)*RHC(1)*Y(  9)
   PL( 13,  7) = GK( 13,  2)*RHC(1)*Y(  2)
   PL( 14,  1) = GK( 14,  1)*RHC(1)*Y(  5)
   PL( 14,  2) = GK( 14,  1)*RHC(1)*Y(  8)
   PL( 14,  6) = GK( 14,  2)*RHC(1)*Y(  6)
   PL( 14,  7) = GK( 14,  2)*RHC(1)*Y(  2)
   PL( 15,  1) = GK( 15,  1)*RHC(1)*Y(  3)
   PL( 15,  2) = GK( 15,  1)*RHC(1)*Y(  9)
   PL( 15,  6) = GK( 15,  2)*RHC(1)*Y(  5)
   PL( 15,  7) = GK( 15,  2)*RHC(1)*Y(  6)
```

```
   PL( 16,  1) = GK( 16,  1)*RHC(1)*Y(  4)
   PL( 16,  2) = GK( 16,  1)*RHC(1)*Y(  5)
   PL( 16,  6) = GK( 16,  2)*RHC(1)*Y(  2)
   PL( 16,  7) = GK( 16,  2)*RHC(1)*Y(  6)
   PL( 17,  1) = GK( 17,  1)*RHC(1)*Y(  5)
   PL( 17,  2) = GK( 17,  1)*RHC(1)*Y(  5)
   PL( 17,  6) = GK( 17,  2)*RHC(1)*Y(  6)
   PL( 17,  7) = GK( 17,  2)*RHC(1)*Y(  1)
   PL( 18,  1) = GK( 18,  1)*RHC(2)*Y(  4)*Y(  4)
   PL( 18,  2) = 2.*GK( 18,  1)*RHC(2)*Y(  5)*Y(  4)
   PL( 18,  6) = GK( 18,  2)*RHC(2)*Y(  7)*Y(  3)
   PL( 18,  7) = GK( 18,  2)*RHC(2)*Y(  6)*Y(  3)
   PL( 18,  8) = GK( 18,  2)*RHC(2)*Y(  6)*Y(  7)
   PL( 19,  1) = GK( 19,  1)*RHC(1)*Y(  4)
   PL( 19,  2) = GK( 19,  1)*RHC(1)*Y( 11)
   PL( 19,  6) = GK( 19,  2)*RHC(1)*Y( 10)
   PL( 19,  7) = GK( 19,  2)*RHC(1)*Y(  3)
   PL( 20,  1) = GK( 20,  1)*RHC(1)*Y(  1)
   PL( 20,  2) = GK( 20,  1)*RHC(1)*Y( 11)
   PL( 20,  6) = GK( 20,  2)*RHC(1)*Y(  5)
   PL( 20,  7) = GK( 20,  2)*RHC(1)*Y(  6)
   PL( 21,  1) = GK( 21,  1)*RHC(1)*Y(  2)
   PL( 21,  2) = GK( 21,  1)*RHC(1)*Y( 10)
   PL( 21,  6) = GK( 21,  2)*RHC(1)*Y(  8)
   PL( 21,  7) = GK( 21,  2)*RHC(1)*Y(  4)
   PL( 22,  1) = GK( 22,  1)*RHC(1)*Y(  1)
   PL( 22,  2) = GK( 22,  1)*RHC(1)*Y( 10)
   PL( 22,  6) = GK( 22,  2)*RHC(1)*Y(  5)
   PL( 22,  7) = GK( 22,  2)*RHC(1)*Y(  4)
   PL( 23,  1) = GK( 23,  1)*RHC(1)*Y(  1)
   PL( 23,  2) = GK( 23,  1)*RHC(1)*Y( 10)
   PL( 23,  6) = GK( 23,  2)*RHC(1)*Y(  5)
   PL( 23,  7) = GK( 23,  2)*RHC(1)*Y(  8)
   PL( 24,  1) = GK( 24,  1)*RHC(1)*Y(  5)
   PL( 24,  2) = GK( 24,  1)*RHC(1)*Y( 10)
   PL( 24,  6) = GK( 24,  2)*RHC(1)*Y(  6)
   PL( 24,  7) = GK( 24,  2)*RHC(1)*Y(  4)
   PL( 25,  1) = GK( 25,  1)*RHC(1)*Y(  5)
   PL( 25,  2) = GK( 25,  1)*RHC(1)*Y( 10)
   PL( 25,  6) = GK( 25,  2)*RHC(1)*Y(  2)
   PL( 25,  7) = GK( 25,  2)*RHC(1)*Y( 11)
   PL( 26,  1) = GK( 26,  1)*RHC(1)*Y(  5)
   PL( 26,  2) = GK( 26,  1)*RHC(1)*Y( 10)
   PL( 26,  6) = GK( 26,  2)*RHC(1)*Y(  3)
   PL( 26,  7) = GK( 26,  2)*RHC(1)*Y(  8)
   PL( 27,  1) = GK( 27,  1)*RHC(1)*Y(  5)
   PL( 27,  2) = GK( 27,  1)*RHC(1)*Y(  8)
   PL( 27,  6) = GK( 27,  2)*RHC(1)*Y(  4)
   PL( 27,  7) = GK( 27,  2)*RHC(1)*Y(  5)
   PL( 28,  1) = GK( 28,  1)*RHC(1)*X
   PL( 28,  6) = GK( 28,  2)*RHC(2)*Y(  2)*X
   PL( 28,  7) = GK( 28,  2)*RHC(2)*Y(  4)*X
   PL( 28,  5) = GK( 28,  1)*RHC(1)*Y(  5)
   PL( 28, 10) = GK( 28,  2)*RHC(2)*Y(  4)*Y(  2)
   C(  1) = Y(  1)
   C(  2) = Y(  2)
   C(  3) = X - C(  1) - C(  2)
   PL( 29,  1) = GK( 29,  1)*RHC(1)*C(  3)
   PL( 29,  6) = 2.*GK( 29,  2)*RHC(2)*Y(  2)*C(  1)
```

```
PL( 29,  5) = CK( 29,  1)*RHC(1)*Y(  1)
PL( 29, 10) = CK( 29,  2)*RHC(2)*Y(  2)*Y(  2)
PL( 30,  1) = CK( 30,  1)*RHC(1)*C(  2)
PL( 30,  6) = 2.*CK( 30,  2)*RHC(2)*Y(  2)*C(  2)
PL( 30,  5) = CK( 30,  1)*RHC(1)*Y(  1)
PL( 30, 10) = CK( 30,  2)*RHC(2)*Y(  2)*Y(  2)
PL( 31,  1) = CK( 31,  1)*RHC(1)*C(  3)
PL( 31,  6) = 2.*CK( 31,  2)*RHC(2)*Y(  2)*C(  3)
PL( 31,  5) = CK( 31,  1)*RHC(1)*Y(  1)
PL( 31, 10) = CK( 31,  2)*RHC(2)*Y(  2)*Y(  2)
C(  1) = Y(  3)
C(  2) = Y(  4)
C(  3) = X - C(  1) - C(  2)
PL( 32,  1) = CK( 32,  1)*RHC(1)*C(  1)
PL( 32,  6) = 2.*CK( 32,  2)*RHC(2)*Y(  4)*C(  1)
PL( 32,  5) = CK( 32,  1)*RHC(1)*Y(  3)
PL( 32, 10) = CK( 32,  2)*RHC(2)*Y(  4)*Y(  4)
PL( 33,  1) = CK( 33,  1)*RHC(1)*C(  2)
PL( 33,  6) = 2.*CK( 33,  2)*RHC(2)*Y(  4)*C(  2)
PL( 33,  5) = CK( 33,  1)*RHC(1)*Y(  3)
PL( 33, 10) = CK( 33,  2)*RHC(2)*Y(  4)*Y(  4)
PL( 34,  1) = CK( 34,  1)*RHC(1)*C(  3)
PL( 34,  6) = 2.*CK( 34,  2)*RHC(2)*Y(  4)*C(  3)
PL( 34,  5) = CK( 34,  1)*RHC(1)*Y(  3)
PL( 34, 10) = CK( 34,  2)*RHC(2)*Y(  4)*Y(  4)
C(  1) = Y(  3)
C(  2) = Y(  1)
C(  3) = Y(  5)
C(  4) = X - C(  1) - C(  2) - C(  3)
PL( 35,  1) = CK( 35,  1)*RHC(1)*C(  1)
PL( 35,  6) = CK( 35,  2)*RHC(2)*Y(  7)*C(  1)
PL( 35,  7) = CK( 35,  2)*RHC(2)*Y( 11)*C(  1)
PL( 35,  5) = CK( 35,  1)*RHC(1)*Y(  3)
PL( 35, 10) = CK( 35,  2)*RHC(2)*Y( 11)*Y(  7)
PL( 36,  1) = CK( 36,  1)*RHC(1)*C(  2)
PL( 36,  6) = CK( 36,  2)*RHC(2)*Y(  7)*C(  2)
PL( 36,  7) = CK( 36,  2)*RHC(2)*Y( 11)*C(  2)
PL( 36,  5) = CK( 36,  1)*RHC(1)*Y(  3)
PL( 36, 10) = CK( 36,  2)*RHC(2)*Y( 11)*Y(  7)
PL( 37,  1) = CK( 37,  1)*RHC(1)*C(  3)
PL( 37,  6) = CK( 37,  2)*RHC(2)*Y(  7)*C(  3)
PL( 37,  7) = CK( 37,  2)*RHC(2)*Y( 11)*C(  3)
PL( 37,  5) = CK( 37,  1)*RHC(1)*Y(  3)
PL( 37, 10) = CK( 37,  2)*RHC(2)*Y( 11)*Y(  7)
PL( 38,  1) = CK( 38,  1)*RHC(1)*C(  4)
PL( 38,  6) = CK( 38,  2)*RHC(2)*Y(  7)*C(  4)
PL( 38,  7) = CK( 38,  2)*RHC(2)*Y( 11)*C(  4)
PL( 38,  5) = CK( 38,  1)*RHC(1)*Y(  3)
PL( 38, 10) = CK( 38,  2)*RHC(2)*Y( 11)*Y(  7)
C(  1) = Y(  3)
C(  2) = Y(  1)
C(  3) = Y(  5)
C(  4) = X - C(  1) - C(  2) - C(  3)
PL( 39,  1) = CK( 39,  1)*RHC(1)*C(  1)
PL( 39,  6) = CK( 39,  2)*RHC(2)*Y(  7)*C(  1)
PL( 39,  7) = CK( 39,  2)*RHC(2)*Y(  9)*C(  1)
PL( 39,  5) = CK( 39,  1)*RHC(1)*Y(  1)
PL( 39, 10) = CK( 39,  2)*RHC(2)*Y(  9)*Y(  7)
PL( 40,  1) = CK( 40,  1)*RHC(1)*C(  2)
```

```
PL( 40,  6) = CK( 40,  2)*RHC(2)*Y(  7)*C(  2)
PL( 40,  7) = CK( 40,  2)*RHC(2)*Y(  9)*C(  2)
PL( 40,  5) = CK( 40,  1)*RHC(1)*Y(  1)
PL( 40, 10) = CK( 40,  2)*RHC(2)*Y(  9)*Y(  7)
PL( 41,  1) = CK( 41,  1)*RHC(1)*C(  3)
PL( 41,  6) = CK( 41,  2)*RHC(2)*Y(  7)*C(  3)
PL( 41,  7) = CK( 41,  2)*RHC(2)*Y(  9)*C(  3)
PL( 41,  5) = CK( 41,  1)*RHC(1)*Y(  1)
PL( 41, 10) = CK( 41,  2)*RHC(2)*Y(  9)*Y(  7)
PL( 42,  1) = CK( 42,  1)*RHC(1)*C(  4)
PL( 42,  6) = CK( 42,  2)*RHC(2)*Y(  7)*C(  4)
PL( 42,  7) = CK( 42,  2)*RHC(2)*Y(  9)*C(  4)
PL( 42,  5) = CK( 42,  1)*RHC(1)*Y(  1)
PL( 42, 10) = CK( 42,  2)*RHC(2)*Y(  9)*Y(  7)
C(  1) = Y(  3)
C(  2) = Y(  1)
C(  3) = Y(  5)
C(  4) = X - C(  1) - C(  2) - C(  3)
PL( 43,  1) = CK( 43,  1)*RHC(1)*C(  1)
PL( 43,  6) = CK( 43,  2)*RHC(2)*Y(  7)*C(  1)
PL( 43,  7) = CK( 43,  2)*RHC(2)*Y( 10)*C(  1)
PL( 43,  5) = CK( 43,  1)*RHC(1)*Y(  4)
PL( 43, 10) = CK( 43,  2)*RHC(2)*Y( 10)*Y(  7)
PL( 44,  1) = CK( 44,  1)*RHC(1)*C(  2)
PL( 44,  6) = CK( 44,  2)*RHC(2)*Y(  7)*C(  2)
PL( 44,  7) = CK( 44,  2)*RHC(2)*Y( 10)*C(  2)
PL( 44,  5) = CK( 44,  1)*RHC(1)*Y(  4)
PL( 44, 10) = CK( 44,  2)*RHC(2)*Y( 10)*Y(  7)
PL( 45,  1) = CK( 45,  1)*RHC(1)*C(  3)
PL( 45,  6) = CK( 45,  2)*RHC(2)*Y(  7)*C(  3)
PL( 45,  7) = CK( 45,  2)*RHC(2)*Y( 10)*C(  3)
PL( 45,  5) = CK( 45,  1)*RHC(1)*Y(  4)
PL( 45, 10) = CK( 45,  2)*RHC(2)*Y( 10)*Y(  7)
PL( 46,  1) = CK( 46,  1)*RHC(1)*C(  4)
PL( 46,  6) = CK( 46,  2)*RHC(2)*Y(  7)*C(  4)
PL( 46,  7) = CK( 46,  2)*RHC(2)*Y( 10)*C(  4)
PL( 46,  5) = CK( 46,  1)*RHC(1)*Y(  4)
PL( 46, 10) = CK( 46,  2)*RHC(2)*Y( 10)*Y(  7)
C(  1) = Y(  3)
C(  2) = Y(  1)
C(  3) = Y(  5)
C(  4) = X - C(  1) - C(  2) - C(  3)
PL( 47,  1) = CK( 47,  1)*RHC(1)*C(  1)
PL( 47,  6) = CK( 47,  2)*RHC(2)*Y(  7)*C(  1)
PL( 47,  7) = CK( 47,  2)*RHC(2)*Y(  8)*C(  1)
PL( 47,  5) = CK( 47,  1)*RHC(1)*Y(  2)
PL( 47, 10) = CK( 47,  2)*RHC(2)*Y(  8)*Y(  7)
PL( 48,  1) = CK( 48,  1)*RHC(1)*C(  2)
PL( 48,  6) = CK( 48,  2)*RHC(2)*Y(  7)*C(  2)
PL( 48,  7) = CK( 48,  2)*RHC(2)*Y(  8)*C(  2)
PL( 48,  5) = CK( 48,  1)*RHC(1)*Y(  2)
PL( 48, 10) = CK( 48,  2)*RHC(2)*Y(  8)*Y(  7)
PL( 49,  1) = CK( 49,  1)*RHC(1)*C(  3)
PL( 49,  6) = CK( 49,  2)*RHC(2)*Y(  7)*C(  3)
PL( 49,  7) = CK( 49,  2)*RHC(2)*Y(  8)*C(  3)
PL( 49,  5) = CK( 49,  1)*RHC(1)*Y(  2)
PL( 49, 10) = CK( 49,  2)*RHC(2)*Y(  8)*Y(  7)
PL( 50,  1) = CK( 50,  1)*RHC(1)*C(  4)
PL( 50,  6) = CK( 50,  2)*RHC(2)*Y(  7)*C(  4)
PL( 50,  7) = GK( 50,  2)*RHC(2)*Y(  6)*C(  4)
PL( 50,  9) = GK( 50,  1)*RHC(1)*Y(  2)
PL( 50, 10) = GK( 50,  2)*RHC(2)*Y(  6)*Y(  7)
C(  1) = Y(  3)
C(  2) = Y(  1)
C(  3) = Y(  5)
C(  4) = X - C(  1) - C(  2) - C(  3)
PL( 51,  1) = GK( 51,  1)*RHC(1)*C(  1)
PL( 51,  6) = GK( 51,  2)*RHC(2)*Y(  7)*C(  1)
PL( 51,  7) = GK( 51,  2)*RHC(2)*Y(  6)*C(  1)
PL( 51,  9) = GK( 51,  1)*RHC(1)*Y(  5)
PL( 51, 10) = GK( 51,  2)*RHC(2)*Y(  6)*Y(  7)
PL( 52,  1) = GK( 52,  1)*RHC(1)*C(  2)
PL( 52,  6) = GK( 52,  2)*RHC(2)*Y(  7)*C(  2)
PL( 52,  7) = GK( 52,  2)*RHC(2)*Y(  6)*C(  2)
PL( 52,  9) = GK( 52,  1)*RHC(1)*Y(  5)
PL( 52, 10) = GK( 52,  2)*RHC(2)*Y(  6)*Y(  7)
PL( 53,  1) = GK( 53,  1)*RHC(1)*C(  3)
PL( 53,  6) = GK( 53,  2)*RHC(2)*Y(  7)*C(  3)
PL( 53,  7) = GK( 53,  2)*RHC(2)*Y(  6)*C(  3)
PL( 53,  9) = GK( 53,  1)*RHC(1)*Y(  5)
PL( 53, 10) = GK( 53,  2)*RHC(2)*Y(  6)*Y(  7)
PL( 54,  1) = GK( 54,  1)*RHC(1)*C(  4)
PL( 54,  6) = GK( 54,  2)*RHC(2)*Y(  7)*C(  4)
PL( 54,  7) = GK( 54,  2)*RHC(2)*Y(  6)*C(  4)
PL( 54,  9) = GK( 54,  1)*RHC(1)*Y(  5)
PL( 54, 10) = GK( 54,  2)*RHC(2)*Y(  6)*Y(  7)
BB=   PL(  2,  1)+   PL(  3,  2)+   PL(  9,  2)+   PL( 11,  2)
BB=BB+   PL( 13,  2)+   PL( 17,  6)+   PL( 20,  2)+   PL( 22,  2)
BB=BB+   PL( 23,  2)+   PL( 25,  1)+   PL( 30,  1)+   PL( 31,  1)
BB=BB+   PL( 29,  5)+   PL( 35,  1)+   PL( 40,  1)+   PL( 41,  1)
AA=   PL( 29, 10)+   PL( 40, 10)
BB=BB+   PL( 42,  1)+   PL( 40,  5)
A(  1,  1) = AA - BB
AA=   PL(  2,  2)+   PL( 11,  7)+   PL( 13,  6)+   PL( 29,  6)
AA=AA+   PL( 30,  6)+   PL( 31,  6)+   PL( 30, 10)+   PL( 42, 10)
BB=   PL( 30,  5)+   PL( 42,  5)
A(  1,  2) = AA - BB
AA=   PL(  9,  6)+   PL( 31, 10)+   PL( 35, 10)
BB=   PL(  3,  5)+   PL( 31,  5)+   PL( 35,  5)
A(  1,  3) = AA - BB
AA=   PL( 22,  6)+   PL( 31, 10)+   PL( 42, 10)
BB=   PL(  2,  6)+   PL( 31,  5)+   PL( 42,  5)
A(  1,  4) = AA - BB
AA=   PL(  2,  1)+   PL(  3,  6)+   PL( 17,  2)+   PL( 20,  7)
AA=AA+   PL( 23,  7)+   PL( 31, 10)+   PL( 41, 10)
BB=   PL( 31,  5)+   PL( 41,  5)
A(  1,  5) = AA - BB
AA=   PL( 11,  6)+   PL( 20,  4)+   PL( 31, 10)+   PL( 42, 10)
BB=   PL( 17,  7)+   PL( 31,  5)+   PL( 42,  5)
A(  1,  6) = AA - BB
AA=   PL( 31, 10)+   PL( 35,  7)+   PL( 40,  7)+   PL( 41,  7)
AA=AA+   PL( 42,  1)+   PL( 42, 10)
BB=   PL( 31,  5)+   PL( 42,  5)
A(  1-  7) = AA - BB
AA=   PL( 23,  6)+   PL( 31, 10)+   PL( 42, 10)
BB=   PL( 13,  1)+   PL( 31,  5)+   PL( 42,  5)
A(  1,  8) = AA - BB
AA=   PL(  5,  7)+   PL( 13,  1)+   PL( 17,  1)+   PL( 22,  7)
```

```
AA=AA+   PL( 31, 10)+   PL( 35,  6)+   PL( 40,  6)+   PL( 41,  6)
AA=AA+   PL( 42,  6)+   PL( 42, 10)
BB=   PL( 31,  5)+   PL( 42,  5)
A(  1,  9) = AA - BB
BB=   PL( 11,  1)+   PL( 22,  1)+   PL( 23,  1)+   PL( 31,  5)
AA=   PL( 31, 10)+   PL( 42, 10)
BB=BB+   PL( 42,  5)
A(  1, 10) = AA - BB
AA=   PL( 31, 10)+   PL( 42, 10)
BB=   PL(  9,  1)+   PL( 20,  1)+   PL( 31,  5)+   PL( 42,  5)
A(  1, 11) = AA - BB
AA=   PL(  2,  2)+   PL( 11,  2)+   PL( 13,  2)+   PL( 28,  5)
AA=AA+2.*PL( 29,  1)+2.*PL( 30,  1)+2.*PL( 31,  1)+2.*PL( 29,  5)
AA=AA+   PL( 48, 10)
BB=   PL( 28, 10)+2.*PL( 29, 10)+   PL( 48,  5)
A(  2,  1) = AA - BB
BB=   PL(  1,  1)+   PL(  2,  2)+   PL(  4,  2)+2.*PL(  6,  1)
BB=BB+   PL(  7,  2)+   PL(  8,  2)+   PL( 11,  7)+   PL( 13,  6)
BB=BB+   PL( 14,  6)+   PL( 16,  7)+   PL( 21,  2)+   PL( 25,  7)
BB=BB+   PL( 28,  7)+   PL( 28, 10)+2.*PL( 29,  6)+2.*PL( 30,  6)
BB=BB+2.*PL( 31,  6)+2.*PL( 30, 10)+   PL( 47,  1)+   PL( 48,  1)
AA=   PL( 28,  5)+2.*PL( 30,  5)+   PL( 50, 10)
BB=BB+   PL( 49,  1)+   PL( 50,  1)+   PL( 50,  5)
A(  2,  2) = AA - BB
AA=   PL(  7,  6)+   PL( 28,  5)+2.*PL( 31,  5)+   PL( 47, 10)
BB=   PL(  1,  2)+   PL( 28, 10)+2.*PL( 31, 10)+   PL( 47,  5)
A(  2,  3) = AA - BB
AA=   PL(  1,  7)+   PL(  2,  6)+   PL(  3,  7)+   PL( 16,  2)
BB=   PL(  4,  1)+   PL( 20,  6)+   PL( 28, 10)+2.*PL( 31, 10)
AA=AA+   PL( 21,  6)+   PL( 28,  5)+2.*PL( 31,  5)+   PL( 50, 10)
BB=BB+   PL( 50,  5)
A(  2,  4) = AA - BB
AA=   PL(  1,  6)+   PL( 14,  2)+   PL( 25,  2)+   PL( 26,  1)
AA=AA+   PL( 28,  5)+2.*PL( 31,  5)+   PL( 45, 10)
BB=   PL(  2,  1)+   PL( 28, 10)+2.*PL( 31, 10)+   PL( 45,  5)
A(  2,  5) = AA - BB
BB=   PL( 11,  6)+   PL( 14,  7)+   PL( 16,  6)+   PL( 28, 10)
AA=   PL(  4,  6)+   PL(  8,  6)+   PL( 28,  5)+2.*PL( 31,  5)
AA=AA+   PL( 50, 10)
BB=BB+2.*PL( 31, 10)+   PL( 50,  5)
A(  2,  6) = AA - BB
AA=   PL(  4,  7)+2.*PL(  6,  7)+   PL( 28,  5)+2.*PL( 31,  5)
AA=AA+   PL( 47,  7)+   PL( 48,  7)+   PL( 49,  7)+   PL( 50,  7)
AA=AA+   PL( 50, 10)
BB=   PL( 29, 10)+2.*PL( 31, 10)+   PL( 50,  5)
A(  2,  7) = AA - BB
AA=   PL(  7,  7)+   PL( 13,  1)+   PL( 14,  1)+   PL( 21,  7)
AA=AA+   PL( 28,  5)+2.*PL( 31,  5)+   PL( 47,  6)+   PL( 48,  6)
AA=AA+   PL( 49,  6)+   PL( 50,  6)+   PL( 50, 10)
BB=   PL( 28, 10)+2.*PL( 31, 10)+   PL( 50,  5)
A(  2,  8) = AA - BB
AA=2.*PL(  6,  6)+   PL( 16,  1)+   PL( 28,  5)+2.*PL( 31,  5)
AA=AA+   PL( 50, 10)
BB=   PL( 13,  7)+   PL( 28, 10)+2.*PL( 31, 10)+   PL( 50,  5)
A(  2,  9) = AA - BB
AA=   PL( 11,  1)+   PL( 25,  1)+   PL( 28,  5)+2.*PL( 31,  5)
AA=AA+   PL( 50, 10)
BB=   PL( 21,  1)+   PL( 26, 10)+2.*PL( 31, 10)+   PL( 50,  5)
A(  2, 10) = AA - BB
```

```
BB=   PL(  7,  1)+   PL(  8,  1)+   PL( 25,  6)+   PL( 28, 10)
AA=   PL( 28,  5)+2.*PL( 30,  5)+   PL( 50, 10)
BB=BB+2.*PL( 31, 10)+   PL( 50,  5)
A(  2, 11) = AA - BB
AA=   PL(  5,  2)+   PL( 34, 10)+   PL( 38, 10)
BB=   PL(  3,  2)+   PL( 34,  5)+   PL( 34,  5)
A(  3,  1) = AA - BB
AA=   PL(  7,  2)+   PL( 34, 10)+   PL( 38, 10)
BB=   PL(  1,  1)+   PL( 34,  5)+   PL( 38,  5)
A(  3,  2) = AA - BB
BB=   PL(  1,  2)+   PL(  3,  1)+   PL(  7,  6)+   PL(  9,  6)
BB=BB+   PL( 10,  6)+   PL( 12,  2)+   PL( 15,  2)+   PL( 18,  8)
BB=BB+   PL( 19,  6)+   PL( 26,  7)+   PL( 32,  1)+   PL( 33,  1)
BB=BB+   PL( 34,  1)+   PL( 32,  5)+   PL( 35,  1)+   PL( 36,  1)
AA=   PL( 32, 10)+   PL( 35, 10)
BB=BB+   PL( 37,  1)+   PL( 38,  1)+   PL( 35,  5)
A(  3,  3) = AA - BB
AA=   PL(  1,  7)+   PL( 12,  7)+   PL( 18,  2)+   PL( 19,  2)
AA=AA+   PL( 32,  6)+   PL( 33,  6)+   PL( 34,  6)+   PL( 33, 10)
AA=AA+   PL( 38, 10)
BB=   PL( 33,  5)+   PL( 38,  5)
A(  3,  4) = AA - BB
AA=   PL(  1,  6)+   P[illegible]  6)+   PL( 10,  [illegible]+   PL( 15,  [illegible]
AA=AA+   PL( 18,  1)+   [illegible] 20,  2)+   PL( 34, 10)+   PL( 37, 10)
BB=   PL( 34,  5)+   PL( 37,  5)
A(  3,  5) = AA - BB
AA=   PL( 12,  6)+   PL( 15,  6[illegible]PL( 34, 10)+   PL( 38, [illegible]
BB=   PL( 10,  7)+   PL( 16, [illegible]   PL( 34,  5)+   PL( 38,  5)
A(  3,  6) = AA - BB
AA=   PL( 34, 10)+   PL( 35,  7)+   PL( 36,  7)+   PL( 37,  7)
AA=AA+   PL( 38,  7)+   PL( 38, [illegible])
BB=   PL( 18,  7)+   PL( 34,  5)+   PL( 38,  5)
A(  3,  7) = AA - BB
BB=   PL(  7,  7)+   PL( 12,  [illegible])+   PL( 26,  6)+   PL( 34,  5)
AA=   PL( 34, 10)+   PL( 38, 10)
BB=BB+   PL( 38,  5)
A(  3,  8) = AA - BB
AA=   PL( 34, 10)+   PL( 38, 10)
BB=   PL(  5,  7)+   PL( 15,  1)+   PL( 34,  5)+   PL( 38,  5)
A(  3,  9) = AA - BB
AA=   PL( 26,  1)+   PL( 34, 10)+   PL( 38, 10)
BB=   PL( 19,  7)+   PL( 34,  5)+   PL( 38,  5)
A(  3, 10) = AA - BB
AA=   PL(  7,  2)+   PL(  5,  1)+   PL( 10,  1)+   PL( 19,  1)
AA=AA+   PL( 34, 10)+   PL( 35,  6)+   PL( 36,  6)+   PL( 37,  6)
AA=AA+   PL( 38,  6)+   PL( 38, 10)
BB=   PL( 24,  5)+   PL( 38,  5)
A(  3, 11) = AA - BB
AA=   PL( 22,  2)+   PL( 28,  5)+2.*PL( 34,  5)+   PL( 44, 10)
BB=   PL(  2,  7)+   PL( 28, 10)+2.*PL( 34, 10)+   PL( 44,  5)
A(  4,  1) = AA - BB
AA=   PL(  1,  1)+   PL(  2,  2)+   PL(  8,  2)+   PL( 16,  7)
BB=   PL(  4,  2)+   PL( 28,  7)+   PL( 28, 10)+2.*PL( 34, 10)
AA=AA+   PL( 21,  2)+   PL( 28,  5)+2.*PL( 34,  5)+   PL( 46, 10)
BB=BB+   PL( 46,  5)
A(  4,  2) = AA - BB
AA=   PL(  1,  2)+   PL( 12,  2)+2.*PL( 18,  8)+   PL( 19,  6)
AA=AA+   PL( 28,  5)+2.*PL( 32,  1)+2.*PL( 33,  1)+2.*PL( 34,  1)
BB=BB+2.*PL( 32,  5)+   PL( 43, 10)
```

```
BB=   PL( 28, 10)+2.*PL( 32, 10)+   PL( 43,  5)
A(  4,  3) = AA - BB
BB=   PL(  1,  1)+   PL(  2,  6)+   PL(  4,  1)+2.*PL(  5,  1)
BB=BB+   PL(  8,  7)+   PL( 12,  1)+   PL( 16,  2)+2.*PL( 18,  2)
BB=BB+   PL( 19,  2)+   PL( 21,  6)+   PL( 22,  6)+   PL( 24,  6)
BB=BB+   PL( 27,  7)+   PL( 28,  6)+   PL( 28, 10)+2.*PL( 32,  6)
BB=BB+2.*PL( 33,  6)+2.*PL( 34,  6)+2.*PL( 33, 10)+   PL( 43,  1)
AA=   PL( 28,  5)+2.*PL( 33,  5)+   PL( 46, 10)
BB=BB+   PL( 44,  1)+   PL( 45,  1)+   PL( 46,  1)+   PL( 46,  5)
A(  4,  4) = AA - BB
AA=   PL(  2,  1)+   PL( 24,  2)+   PL( 27,  2)+   PL( 28,  1)
BB=   PL(  1,  4)+2.*PL( 18,  1)+   PL( 28, 10)+2.*PL( 34, 10)
AA=AA+   PL( 28,  5)+2.*PL( 34,  5)+   PL( 45, 10)
BB=BB+   PL( 45,  5)
A(  4,  5) = AA - BB
AA=   PL(  4,  6)+   PL( 16,  6)+2.*PL( 18,  6)+   PL( 28,  5)
BB=   PL(  6,  6)+   PL( 12,  6)+   PL( 24,  7)+   PL( 28, 10)
AA=AA+2.*PL( 34,  5)+   PL( 44, 10)
BB=BB+2.*PL( 34, 10)+   PL( 46,  5)
A(  4,  6) = AA - BB
AA=   PL(  4,  7)+2.*PL(  5,  7)+2.*PL( 18,  7)+   PL( 28,  5)
AA=AA+2.*PL( 34,  5)+   PL( 43,  7)+   PL( 4 ,  7)+   PL( 45,  1)
AA=AA+   PL( 46,  7)+   PL( 46, 10)
BB=   PL( 28, 10)+2.*PL( 34, 10)+   PL( 46,  5)
A(  4,  7) = AA - BB
AA=   PL( 12,  1)+   PL( 27,  1)+   PL( 28,  5)+2.*PL( 34,  5)
AA=AA+   PL( 46, 10)
BB=   PL( 21,  7)+   PL( 28, 10)+2.*PL( 34, 10)+   PL( 46,  5)
A(  4,  8) = AA - BB
BB=   PL( 16,  1)+   PL( 22,  7)+   PL( 27,  4)+   PL( 28, 10)
AA=   PL( 28,  5)+2.*PL( 34,  5)+   PL( 46, 10)
BB=BB+2.*PL( 34, 10)+   PL( 46,  5)
A(  4,  9) = AA - BB
AA=   PL( 19,  7)+   PL( 21,  1)+   PL( 22,  1)+   PL( 24,  1)
AA=AA+   PL( 28,  5)+2.*PL( 34,  5)+   PL( 43,  6)+   PL( 44,  6)
AA=AA+   PL( 45,  6)+   PL( 46,  6)+   PL( 46, 10)
BB=   PL( 28, 10)+2.*PL( 34, 10)+   PL( 46,  5)
A(  4, 10) = AA - BB
AA=2.*PL(  5,  6)+   PL(  8,  1)+   PL( 28,  5)+2.*PL( 34,  5)
AA=AA+   PL( 46, 10)
BB=   PL( 19,  1)+   PL( 28, 10)+2.*PL( 34, 10)+   PL( 46,  5)
A(  4, 11) = AA - BB
AA=   PL(  2,  7)+2.*PL(  3,  2)+   PL( 17,  6)+   PL( 20,  2)
AA=AA+   PL( 23,  2)+   PL( 28, 10)+   PL( 52, 10)
BB=   PL( 28,  5)+   PL( 52,  5)
A(  5,  1) = AA - BB
AA=   PL(  1,  1)+   PL( 14,  6)+   PL( 25,  1)+   PL( 28,  7)
AA=AA+   PL( 20, 10)+   PL( 54, 10)
BB=   PL(  7,  2)+   PL( 26,  5)+   PL( 54,  5)
A(  5,  2) = AA - BB
AA=   PL(  1,  2)+2.*PL(  3,  1)+   PL( 10,  6)+   PL( 15,  2)
AA=AA+   PL( 18,  8)+   PL( 26,  7)+   PL( 28, 10)+   PL( 51, 10)
BB=   PL( 28,  5)+   PL( 51,  5)
A(  5,  3) = AA - BB
AA=   PL(  2,  6)+   PL( 24,  6)+   PL( 27,  7)+   PL( 28,  6)
AA=AA+   PL( 28, 10)+   PL( 54, 10)
BB=   PL(  1,  7)+   PL( 18,  2)+   PL( 28,  5)+   PL( 54,  5)
A(  5,  4) = AA - BB
BB=   PL(  1,  6)+   PL(  2,  1)+2.*PL(  3,  6)+   PL( 16,  2)
```

```
BB=BB+   PL( 14,  2)+   PL( 15,  7)+   PL( 17,  2)+   PL( 18,  1)
BB=BB+   PL( 20,  7)+   PL( 23,  7)+   PL( 24,  2)+   PL( 25,  2)
BB=BB+   PL( 26,  2)+   PL( 27,  2)+   PL( 28,  1)+   PL( 28,  5)
BB=BB+   PL( 51,  1)+   PL( 52,  1)+   PL( 53,  1)+   PL( 54,  1)
AA=   PL( 28, 10)+   PL( 53, 10)
BB=BB+   PL( 53,  5)
A(  5,  5) = AA - BB
AA=   PL( 10,  7)+   PL( 14,  7)+   PL( 17,  7)+   PL( 18,  6)
AA=AA+   PL( 24,  7)+   PL( 28, 10)+   PL( 51,  6)+   PL( 52,  6)
AA=AA+   PL( 53,  6)+   PL( 54,  6)+   PL( 54, 10)
BB=   PL( 15,  6)+   PL( 20,  6)+   PL( 28,  5)+   PL( 54,  5)
A(  5,  6) = AA - BB
AA=   PL( 18,  7)+   PL( 28, 10)+   PL( 51,  7)+   PL( 52,  7)
AA=AA+   PL( 53,  7)+   PL( 54,  7)+   PL( 54, 10)
BB=   PL( 28,  5)+   PL( 54,  5)
A(  5,  7) = AA - BB
BB=   PL( 14,  1)+   PL( 23,  6)+   PL( 27,  1)+   PL( 28,  5)
AA=   PL( 26,  6)+   PL( 28, 10)+   PL( 54, 10)
BB=BB+   PL( 54,  5)
A(  5,  8) = AA - BB
AA=   PL( 13,  1)+   PL( 27,  6)+   PL( 28, 10)+   PL( 54, 10)
BB=   PL( 17,  1)+   PL( 28,  5)+   PL( 54,  5)
A(  5,  9) = AA - BB
BB=   PL( 24,  1)+   PL( 25,  1)+   PL( 26,  1)+   PL( 28,  5)
AA=   PL( 23,  1)+   PL( 28, 10)+   PL( 54, 10)
BB=BB+   PL( 54,  5)
A(  5, 10) = AA - BB
AA=   PL( 20,  1)+   PL( 25,  4)+   PL( 28, 10)+   PL( 54, 10)
BB=   PL( 10,  1)+   PL( 28,  5)+   PL( 54,  5)
A(  5, 11) = AA - BB
AA=   PL( 11,  2)+   PL( 20,  2)+   PL( 52,  5)
BB=   PL( 17,  6)+   PL( 52, 10)
A(  6,  1) = AA - BB
AA=   PL(  4,  2)+   PL(  8,  2)+   PL( 54,  5)
BB=   PL( 11,  7)+   PL( 14,  6)+   PL( 16,  7)+   PL( 54, 10)
A(  6,  2) = AA - BB
AA=   PL( 12,  2)+   PL( 15,  2)+   PL( 51,  5)
BB=   PL( 10,  6)+   PL( 18,  8)+   PL( 51, 10)
A(  6,  3) = AA - BB
AA=   PL(  4,  1)+   PL( 16,  2)+   PL( 18,  2)+   PL( 54,  5)
BB=   PL(  8,  7)+   PL( 12,  7)+   PL( 24,  6)+   PL( 54, 10)
A(  6,  4) = AA - BB
AA=   PL( 10,  2)+   PL( 14,  2)+   PL( 17,  2)+   PL( 18,  1)
AA=AA+   PL( 24,  2)+   PL( 51,  1)+   PL( 52,  1)+   PL( 53,  1)
AA=AA+   PL( 54,  1)+   PL( 53,  5)
BB=   PL( 15,  7)+   PL( 20,  7)+   PL( 53, 10)
A(  6,  5) = AA - BB
BB=   PL(  4,  6)+   PL(  8,  6)+   PL( 10,  7)+   PL( 11,  6)
BB=BB+   PL( 12,  6)+   PL( 14,  7)+   PL( 15,  6)+   PL( 16,  6)
BB=BB+   PL( 17,  7)+   PL( 18,  6)+   PL( 20,  6)+   PL( 24,  7)
BB=BB+   PL( 51,  6)+   PL( 52,  6)+   PL( 53,  6)+   PL( 54,  6)
AA=   PL( 54,  5)
BB=BB+   PL( 54, 10)
A(  6,  6) = AA - BB
BB=   PL(  4,  7)+   PL( 12,  7)+   PL( 51,  7)+   PL( 52,  7)
AA=   PL( 54,  5)
BB=BB+   PL( 53,  7)+   PL( 54,  7)+   PL( 54, 10)
A(  6,  7) = AA - BB
AA=   PL( 12,  1)+   PL( 14,  1)+   PL( 54,  5)
```

```
BB=   PL( 54, 10)
A(  6,  8) = AA - BB
AA=   PL( 15,  1)+   PL( 16,  1)+   PL( 17,  1)+   PL( 54,  5)
BB=   PL( 54, 10)
A(  6,  9) = AA - BB
AA=   PL( 11,  1)+   PL( 24,  1)+   PL( 54,  5)
BB=   PL( 54, 10)
A(  6, 10) = AA - BB
AA=   PL(  8,  1)+   PL( 10,  1)+   PL( 20,  1)+   PL( 54,  5)
BB=   PL( 54, 10)
A(  6, 11) = AA - BB
AA=   PL( 38,  5)+   PL( 35,  1)+   PL( 40,  1)+   PL( 41,  1)
AA=AA+   PL( 42,  1)+   PL( 40,  5)+   PL( 44,  5)+   PL( 48,  5)
BB=   PL( 36, 10)+   PL( 40, 10)+   PL( 44, 10)+   PL( 48, 10)
AA=AA+   PL( 52,  5)
BB=BB+   PL( 52, 10)
A(  7,  1) = AA - BB
AA=   PL(  4,  2)+   PL(  6,  1)+   PL( 38,  5)+   PL( [illegible],  5)
AA=AA+   PL( 46,  5)+   PL( 47,  1)+   PL( 48,  1)+   PL( 44,  1)
BB=   PL( 38, 10)+   PL( 42, 10)+   PL( 46, 10)+   PL( 50, 10)
AA=AA+   PL( 50,  1)+   PL( 50,  5)+   PL( 54,  5)
BB=BB+   PL( 54, 10)
A(  7,  2) = AA - BB
AA=   PL( 35,  1)+   PL( 36,  1)+   PL( 37,  1)+   PL( 38,  1)
BB=   PL( 18,  8)+   PL( 35, 10)+   PL( 39, 10)+   PL( 43, 10)
AA=AA+   PL( 35,  5)+   PL( 39,  5)+   PL( 43,  5)+   PL( 47,  5)
AA=AA+   PL( 51,  5)
BB=BB+   PL( 47, 10)+   PL( 51, 10)
A(  7,  3) = AA - BB
AA=   PL(  4,  1)+   PL(  5,  1)+   PL( 18,  2)+   PL( 30,  5)
AA=AA+   PL( 42,  5)+   PL( 43,  1)+   PL( 44,  1)+   PL( 45,  1)
BB=   PL( 38, 10)+   PL( 42, 10)+   PL( 46, 10)+   PL( 50, 10)
AA=AA+   PL( 46,  1)+   PL( 46,  5)+   PL( 50,  5)+   PL( 54,  5)
BB=BB+   PL( 54, 10)
A(  7,  4) = AA - BB
AA=   PL( 18,  1)+   PL( 37,  5)+   PL( 41,  5)+   PL( 45,  5)
AA=AA+   PL( 49,  5)+   PL( 51,  1)+   PL( 52,  1)+   PL( 5[illegible],  1)
BB=   PL( 37, 10)+   PL( 41, 10)+   PL( 45, 10)+   PL( 49, 10)
AA=AA+   PL( 54,  1)+   PL( 53,  5)
BB=BB+   PL( 53, 10)
A(  7,  5) = AA - BB
BB=   PL(  4,  6)+   PL( 18,  6)+   PL( 38, 10)+   PL( 42, 10)
BB=BB+   PL( 46, 10)+   PL( 50, 10)+   PL( 51,  6)+   PL( 52,  6)
AA=   PL( 38,  5)+   PL( 42,  5)+   PL( 46,  5)+   PL( 50, [illegible]
AA=AA+   PL( 54,  5)
BB=BB+   PL( 53,  6)+   PL( 54,  6)+   PL( 54, 10)
A(  7,  6) = AA - BB
BB=   PL(  4,  7)+   PL(  5,  7)+   PL(  6,  7)+   PL( 18,  7)
BB=BB+   PL( 35,  7)+   PL( 36,  7)+   PL( 37,  7)+   PL( 38,  7)
BB=BB+   PL( 38, 10)+   PL( 39, [illegible]   PL( 40,  7)+   PL( 41,  7)
BB=BB+   PL( 42,  7)+   PL( 42[illegible]   PL( 43,  7)+   PL( 44,  7)
BB=BB+   PL( 45,  7)+   PL( 46[illegible]   PL( 46, 10)+   PL( 47,  7)
BB=BB+   PL( 49,  7)+   PL( 49, [illegible]   PL( 50,  7)+   PL( 50, 10)
AA=   PL( 38,  5)+   PL( 42,  5) [illegible]( 46,  5)+   PL( 50,  5)
BB=BB+   PL( 51,  7)+   PL( 52, [illegible]   PL( 53,  7)+   PL( 54,  7)
AA=AA+   PL( 54,  5)
BB=BB+   PL( 54, 10)
A(  7,  7) = AA - BB
BB=   PL( 38, 10)+   PL( 42, 10)+   PL( 46, 10)+   PL( 4[illegible],  6)
```

```
AA=   PL( 38,  5)+   PL( 42,  5)+   PL( 46,  5)+   PL( 50,  5)
BB=BB+   PL( 48,  6)+   PL( 45,  6)+   PL( 50,  6)+   PL( 50, 10)
AA=AA+   PL( 54,  5)
BB=BB+   PL( 54, 10)
A( 7,  8) = AA - BB
BB=   PL(  6,  6)+   PL( 33, 10)+   PL( 35,  6)+   PL( 40,  6)
BB=BB+   PL( 41,  6)+   PL( 42,  6)+   PL( 42, 10)+   PL( 46, 10)
AA=   PL( 38,  5)+   PL( 42,  5)+   PL( 46,  5)+   PL( 50,  5)
AA=AA+   PL( 54,  5)
BB=BB+   PL( 50, 10)+   PL( 54, 10)
A( 7,  9) = AA - BB
BB=   PL( 38, 10)+   PL( 42, 10)+   PL( 43,  6)+   PL( 44,  6)
AA=   PL( 38,  5)+   PL( 42,  5)+   PL( 46,  5)+   PL( 50,  5)
BB=BB+   PL( 45,  6)+   PL( 48,  6)+   PL( 46, 10)+   PL( 50, 10)
AA=AA+   PL( 54,  5)
BB=BB+   PL( 54, 10)
A( 7, 10) = AA - BB
BB=   PL(  5,  6)+   PL( 35,  6)+   PL( 36,  6)+   PL( 37,  6)
BB=BB+   PL( 38,  6)+   PL( 38, 10)+   PL( 42, 10)+   PL( 46, 10)
AA=   PL( 38,  5)+   PL( 42,  5)+   PL( 46,  5)+   PL( 50,  5)
AA=AA+   PL( 54,  5)
BB=BB+   PL( 50, 10)+   PL( 54, 10)
A( 7, 11) = AA - BB
AA=   PL( 23,  2)+   PL( 48,  5)
BB=   PL( 13,  2)+   PL( 48, 10)
A( 8,  1) = AA - BB
AA=   PL(  7,  2)+   PL( 13,  6)+   PL( 14,  6)+   PL( 21,  2)
AA=AA+   PL( 47,  1)+   PL( 48,  1)+   PL( 49,  1)+   PL( 50,  1)
AA=AA+   PL( 50,  5)
BB=   PL( 50, 10)
A( 8,  2) = AA - BB
AA=   PL( 47,  5)
BB=   PL(  7,  6)+   PL( 12,  2)+   PL( 26,  7)+   PL( 47, 10)
A( 8,  3) = AA - BB
AA=   PL( 12,  7)+   PL( 27,  7)+   PL( 50,  5)
BB=   PL( 21,  6)+   PL( 50, 10)
A( 8,  4) = AA - BB
AA=   PL( 26,  2)+   PL( 45,  5)
BB=   PL( 14,  2)+   PL( 23,  7)+   PL( 27,  2)+   PL( 45, 10)
A( 8,  5) = AA - BB
AA=   PL( 12,  6)+   PL( 14,  7)+   PL( 50,  5)
BB=   PL( 50, 10)
A( 8,  6) = AA - BB
BB=   PL( 41,  7)+   PL( 48,  7)+   PL( 49,  7)+   PL( 50,  7)
AA=   PL( 50,  5)
BB=BB+   PL( 50, 10)
A( 8,  7) = AA - BB
BB=   PL(  7,  7)+   PL( 12,  1)+   PL( 13,  1)+   PL( 14,  1)
BB=BB+   PL( 21,  7)+   PL( 23,  6)+   PL( 26,  6)+   PL( 27,  1)
BB=BB+   PL( 47,  6)+   PL( 48,  6)+   PL( 45,  6)+   PL( 50,  6)
AA=   PL( 50,  5)
BB=BB+   PL( 50, 10)
A( 8,  8) = AA - BB
AA=   PL( 13,  7)+   PL( 27,  6)+   PL( 50,  5)
BB=   PL( 50, 10)
A( 8,  9) = AA - BB
AA=   PL( 21,  1)+   PL( 23,  1)+   PL( 26,  1)+   PL( 50,  5)
BB=   PL( 50, 10)
A( 8, 10) = AA - BB
```

```
AA=   PL(  7,  1)+   PL( 50,  5)
BB=   PL( 50, 10)
A( 8, 11) = AA - BB
AA=   PL(  5,  2)+   PL( 13,  2)+   PL( 17,  6)+   PL( 22,  2)
AA=AA+   PL( 39,  1)+   PL( 40,  1)+   PL( 41,  1)+   PL( 42,  1)
AA=AA+   PL( 40,  5)
BB=   PL( 40, 10)
A( 9,  1) = AA - BB
AA=   PL(  6,  1)+   PL( 16,  7)+   PL( 42,  5)
BB=   PL( 13,  6)+   PL( 42, 10)
A( 9,  2) = AA - BB
AA=   PL( 39,  5)
BB=   PL(  9,  6)+   PL( 15,  2)+   PL( 39, 10)
A( 9,  3) = AA - BB
AA=   PL( 42,  5)
BB=   PL( 16,  2)+   PL( 22,  6)+   PL( 27,  1)+   PL( 42, 10)
A( 9,  4) = AA - BB
AA=   PL( 15,  7)+   PL( 27,  2)+   PL( 41,  5)
BB=   PL( 17,  2)+   PL( 41, 10)
A( 9,  5) = AA - BB
AA=   PL( 15,  6)+   PL( 16,  6)+   PL( 17,  7)+   PL( 42,  5)
BB=   PL( 42, 10)
A( 9,  6) = AA - BB
BB=   PL(  6,  7)+   PL( 39,  7)+   PL( 40,  7)+   PL( 41,  7)
AA=   PL( 42,  5)
BB=BB+   PL( 42,  7)+   PL( 42, 10)
A( 9,  7) = AA - BB
AA=   PL( 13,  1)+   PL( 27,  1)+   PL( 42,  5)
BB=   PL( 42, 10)
A( 9,  8) = AA - BB
BB=   PL(  6,  6)+   PL(  5,  7)+   PL( 13,  7)+   PL( 15,  1)
BB=BB+   PL( 16,  1)+   PL( 17,  1)+   PL( 22,  7)+   PL( 27,  6)
BB=BB+   PL( 39,  6)+   PL( 40,  6)+   PL( 41,  6)+   PL( 42,  6)
AA=   PL( 42,  5)
BB=BB+   PL( 42, 10)
A( 9,  9) = AA - BB
AA=   PL( 22,  1)+   PL( 42,  5)
BB=   PL( 42, 10)
A( 9, 10) = AA - BB
AA=   PL(  5,  1)+   PL( 42,  5)
BB=   PL( 42, 10)
A( 9, 11) = AA - BB
AA=   PL( 44,  5)
BB=   PL( 11,  2)+   PL( 22,  2)+   PL( 23,  2)+   PL( 44, 10)
A( 10,  1) = AA - BB
AA=   PL( 11,  7)+   PL( 25,  7)+   PL( 46,  5)
BB=   PL( 21,  2)+   PL( 46, 10)
A( 10,  2) = AA - BB
AA=   PL( 26,  7)+   PL( 43,  5)
BB=   PL( 19,  6)+   PL( 43, 10)
A( 10,  3) = AA - BB
AA=   PL( 19,  2)+   PL( 21,  6)+   PL( 22,  6)+   PL( 24,  6)
AA=AA+   PL( 43,  1)+   PL( 44,  1)+   PL( 45,  1)+   PL( 46,  1)
AA=AA+   PL( 46,  5)
BB=   PL( 46, 10)
A( 10,  4) = AA - BB
AA=   PL( 23,  7)+   PL( 45,  5)
BB=   PL( 24,  2)+   PL( 25,  2)+   PL( 26,  2)+   PL( 45, 10)
A( 10,  5) = AA - BB
```

```
AA=   PL( 11,  6)*   PL( 24,  7)*   PL( 46,  5)
BB=   PL( 46, 10)
A( 10,  6) = AA - BB
BB=   PL( 43,  7)*   PL( 44,  7)*   PL( 45,  7)*   PL( 46,  7)
AA=   PL( 46,  5)
BB=BB*    PL( 46, 10)
A( 10,  7) = AA - BB
AA=   PL( 21,  7)*   PL( 23,  6)*   PL( 26,  6)*   PL( 46,  5)
BB=   PL( 46, 10)
A( 10,  8) = AA - BB
AA=   PL( 22,  7)*   PL( 44,  5)
BB=   PL( 46, 10)
A( 10,  9) = AA - BB
BB=   PL( 11,  1)*   PL( 15,  7)*   PL( 21,  1)*   PL( 22,  1)
BB=BB*    PL( 23,  1)*   PL( 24,  1)*   PL( 25,  1)*   PL( 26,  1)
BB=BB*    PL( 43,  6)*   PL( 44,  6)*   PL( 45,  6)*   PL( 46,  6)
AA=   PL( 46,  5)
BB=BB*    PL( 46, 10)
A( 10, 10) = AA - BB
AA=   PL( 15,  1)*   PL( 25,  6)*   PL( 46,  5)
BB=   PL( 46, 10)
A( 10, 11) = AA - BB
AA=   PL( 36,  5)
BB=   PL(  9,  2)*   PL( 20,  2)*   PL( 36, 10)
A( 11,  1) = AA - BB
AA=   PL( 38,  5)
BB=   PL(  7,  2)*   PL(  8,  2)*   PL( 25,  7)*   PL( 38, 10)
A( 11,  2) = AA - BB
AA=   PL(  7,  6)*   PL(  9,  3)*   PL( 10,  6)*   PL( 19,  6)
AA=AA*    PL( 35,  1)*   PL( 36,  1)*   PL( 37,  1)*   PL( 38,  1)
AA=AA*    PL( 35,  5)
BB=   PL( 35, 10)
A( 11,  3) = AA - BB
AA=   PL(  5,  1)*   PL(  8,  7)*   PL( 38,  5)
BB=   PL( 19,  2)*   PL( 38, 10)
A( 11,  4) = AA - BB
AA=   PL( 20,  7)*   PL( 25,  2)*   PL( 37,  5)
BB=   PL( 10,  2)*   PL( 37, 10)
A( 11,  5) = AA - BB
AA=   PL(  8,  6)*   PL( 16,  7)*   PL( 20,  6)*   PL( 38,  5)
BB=   PL( 38, 10)
A( 11,  6) = AA - BB
BB=   PL(  5,  7)*   PL( 35,  7)*   PL( 36,  7)*   PL( 37,  7)
AA=   PL( 38,  5)
BB=BB*    PL( 38,  7)*   PL( 38, 10)
A( 11,  7) = AA - BB
AA=   PL(  7,  7)*   PL( 38,  5)
BB=   PL( 38, 10)
A( 11,  8) = AA - BB
AA=   PL(  9,  7)*   PL( 38,  5)
BB=   PL( 38, 10)
A( 11,  9) = AA - BB
AA=   PL( 19,  7)*   PL( 25,  1)*   PL( 38,  5)
BB=   PL( 38, 10)
A( 11, 10) = AA - BB
BB=   PL(  5,  6)*   PL(  7,  1)*   PL(  8,  1)*   PL(  9,  1)
BB=BB*    PL( 10,  1)*   PL( 19,  1)*   PL( 20,  1)*   PL( 25,  6)
BB=BB*    PL( 35,  6)*   PL( 36,  6)*   PL( 37,  6)*   PL( 38,  6)
AA=   PL( 38,  5)
```

```
BB=BB*    PL( 38, 10)
A( 11, 11) = AA - BB
PL(  1,  1)=Y(  2)*PL(  1,  1)
PL(  1,  6)=Y(  5)*PL(  1,  6)
PL(  2,  1)=Y(  5)*PL(  2,  1)
PL(  2,  6)=Y(  4)*PL(  2,  6)
PL(  3,  1)=Y(  3)*PL(  3,  1)
PL(  3,  6)=Y(  5)*PL(  3,  6)/2.
PL(  4,  1)=Y(  4)*PL(  4,  1)
PL(  4,  6)=Y(  6)*PL(  4,  6)
PL(  5,  1)=Y(  4)*PL(  5,  1)/2.
PL(  5,  6)=Y( 11)*PL(  5,  6)
PL(  6,  1)=Y(  2)*PL(  6,  1)/2.
PL(  6,  6)=Y(  9)*PL(  6,  6)
PL(  7,  1)=Y( 11)*PL(  7,  1)
PL(  7,  6)=Y(  3)*PL(  7,  6)
PL(  8,  1)=Y( 11)*PL(  8,  1)
PL(  8,  6)=Y(  6)*PL(  8,  6)
PL(  9,  1)=Y( 11)*PL(  9,  1)
PL(  9,  6)=Y(  3)*PL(  9,  6)
PL( 10,  1)=Y( 11)*PL( 10,  1)
PL( 10,  6)=Y(  3)*PL( 10,  6)
PL( 11,  1)=Y( 10)*PL( 11,  1)
PL( 11,  6)=Y(  6)*PL( 11,  6)
PL( 12,  1)=Y( [illegible])*PL( 12,  1)
PL( 12,  6)=Y(  6)*PL( 12,  6)
PL( 13,  1)=Y(  8)*PL( 13,  1)
PL( 13,  6)=Y(  2)*PL( 13,  6)
PL( 14,  1)=Y(  8)*PL( 14,  1)
PL( 14,  6)=Y(  2)*PL( 14,  6)
PL( 15,  1)=Y(  9)*PL( 15,  1)
PL( 15,  6)=Y(  6)*PL( 15,  6)
PL( 16,  1)=Y(  9)*PL( 16,  1)
PL( 16,  6)=Y(  6)*PL( 16,  6)
PL( 17,  1)=Y(  9)*PL( 17,  1)
PL( 17,  6)=Y(  1)*PL( 17,  6)
PL( 18,  1)=Y(  5)*PL( 18,  1)
PL( 18,  6)=Y(  6)*PL( 18,  6)
PL( 19,  1)=Y( 11)*PL( 19,  1)
PL( 19,  6)=Y(  3)*PL( 19,  6)
PL( 20,  1)=Y( 11)*PL( 20,  1)
PL( 20,  6)=Y(  6)*PL( 20,  6)
PL( 21,  1)=Y( 10)*PL( 21,  1)
PL( 21,  6)=Y(  4)*PL( 21,  6)
PL( 22,  1)=Y( 10)*PL( 22,  1)
PL( 22,  6)=Y(  4)*PL( 22,  6)
PL( 23,  1)=Y( 10)*PL( 23,  1)
PL( 23,  6)=Y(  8)*PL( 23,  6)
PL( 24,  1)=Y( 10)*PL( 24,  1)
PL( 24,  6)=Y(  4)*PL( 24,  6)
PL( 25,  1)=Y( 10)*PL( 25,  1)
PL( 25,  6)=Y( 11)*PL( 25,  6)
PL( 26,  1)=Y( 10)*PL( 24,  1)
PL( 26,  6)=Y(  8)*PL( 26,  6)
PL( 27,  1)=Y(  8)*PL( 27,  1)
PL( 27,  6)=Y(  9)*PL( 27,  6)
PL( 28,  1)=Y(  5)*PL( 28,  1)
PL( 28,  6)=Y(  4)*PL( 28,  6)
PL( 29,  1)=Y(  1)*PL( 29,  1)
```

```
      PL( 29, 6)=Y( 2)*PL( 29, 6)/2.
      PL( 30, 1)=Y( 1)*PL( 30, 1)
      PL( 30, 6)=Y( 2)*PL( 30, 6)/2.
      PL( 31, 1)=Y( 1)*PL( 31, 1)
      PL( 31, 6)=Y( 2)*PL( 31, 6)/2.
      PL( 32, 1)=Y( 3)*PL( 32, 1)
      PL( 32, 6)=Y( 4)*PL( 32, 6)/2.
      PL( 33, 1)=Y( 3)*PL( 33, 1)
      PL( 33, 6)=Y( 4)*PL( 33, 6)/2.
      PL( 34, 1)=Y( 3)*PL( 34, 1)
      PL( 34, 6)=Y( 4)*PL( 34, 6)/2.
      PL( 35, 1)=Y( 3)*PL( 35, 1)
      PL( 35, 6)=Y( 11)*PL( 35, 6)
      PL( 36, 1)=Y( 3)*PL( 36, 1)
      PL( 36, 6)=Y( 11)*PL( 36, 6)
      PL( 37, 1)=Y( 3)*PL( 37, 1)
      PL( 37, 6)=Y( 11)*PL( 37, 6)
      PL( 38, 1)=Y( 3)*PL( 38, 1)
      PL( 38, 6)=Y( 11)*PL( 38, 6)
      PL( 39, 1)=Y( 1)*PL( 39, 1)
      PL( 39, 6)=Y( 9)*PL( 39, 6)
      PL( 40, 1)=Y( 1)*PL( 40, 1)
      PL( 40, 6)=Y( 9)*PL( 40, 6)
      PL( 41, 1)=Y( 1)*PL( 41, 1)
      PL( 41, 6)=Y( 9)*PL( 41, 6)
      PL( 42, 1)=Y( 1)*PL( 42, 1)
      PL( 42, 6)=Y( 9)*PL( 42, 6)
      PL( 43, 1)=Y( 4)*PL( 43, 1)
      PL( 43, 6)=Y( 10)*PL( 43, 6)
      PL( 44, 1)=Y( 4)*PL( 44, 1)
      PL( 44, 6)=Y( 10)*PL( 44, 6)
      PL( 45, 1)=Y( 4)*PL( 45, 1)
      PL( 45, 6)=Y( 10)*PL( 45, 6)
      PL( 46, 1)=Y( 4)*PL( 46, 1)
      PL( 46, 6)=Y( 10)*PL( 46, 6)
      PL( 47, 1)=Y( 2)*PL( 47, 1)
      PL( 47, 6)=Y( 8)*PL( 47, 6)
      PL( 48, 1)=Y( 2)*PL( 48, 1)
      PL( 48, 6)=Y( 8)*PL( 48, 6)
      PL( 49, 1)=Y( 2)*PL( 49, 1)
      PL( 49, 6)=Y( 8)*PL( 49, 6)
      PL( 50, 1)=Y( 2)*PL( 50, 1)
      PL( 50, 6)=Y( 8)*PL( 50, 6)
      PL( 51, 1)=Y( 5)*PL( 51, 1)
      PL( 51, 6)=Y( 6)*PL( 51, 6)
      PL( 52, 1)=Y( 5)*PL( 52, 1)
      PL( 52, 6)=Y( 6)*PL( 52, 6)
      PL( 53, 1)=Y( 5)*PL( 53, 1)
      PL( 53, 6)=Y( 6)*PL( 53, 6)
      PL( 54, 1)=Y( 5)*PL( 54, 1)
      PL( 54, 6)=Y( 6)*PL( 54, 6)
   30 CONTINUE
      BB=   PL(  2,  6)+   PL(  3,  1)+   PL(  5,  1)+   PL( 11,  1)
      AA=   PL(  2,  1)+   PL(  3,  6)+   PL(  5,  6)+   PL( 11,  6)
      BB=BB+   PL( 13,  1)+   PL( 17,  6)+   PL( 20,  1)+   PL( 22,  1)
      AA=AA+   PL( 13,  6)+   PL( 17,  1)+   PL( 20,  6)+   PL( 22,  6)
      BB=BB+   PL( 23,  1)+   PL( 29,  1)+   PL( 30,  1)+   PL( 31,  1)
      AA=AA+   PL( 23,  6)+   PL( 29,  6)+   PL( 30,  6)+   PL( 31,  6)
      AA=AA+   PL( 39,  6)+   PL( 40,  6)+   PL( 41,  6)+   PL( 42,  6)
```

```
      BB=BB+   PL( 39,  1)+   PL( 40,  1)+   PL( 41,  1)+   PL( 42,  1)
      B(  1) = AA - BB
      BB=   PL(  1,  1)+   PL(  2,  1)+   PL(  4,  1)+2.*PL(  6,  1)
      AA=   PL(  1,  6)+   PL(  2,  6)+   PL(  4,  6)+2.*PL(  6,  6)
      AA=AA+   PL(  7,  6)+   PL(  8,  6)+   PL( 11,  1)+   PL( 13,  1)
      BB=BB+   PL(  7,  1)+   PL(  8,  1)+   PL( 11,  6)+   PL( 13,  6)
      AA=AA+   PL( 14,  1)+   PL( 16,  1)+   PL( 21,  6)+   PL( 25,  1)
      BB=BB+   PL( 14,  6)+   PL( 16,  6)+   PL( 21,  1)+   PL( 25,  6)
      BB=BB+   PL( 28,  6)+2.*PL( 29,  6)+2.*PL( 30,  6)+2.*PL( 31,  6)
      AA=AA+   PL( 28,  1)+2.*PL( 29,  1)+2.*PL( 30,  1)+2.*PL( 31,  1)
      AA=AA+   PL( 47,  6)+   PL( 48,  6)+   PL( 49,  6)+   PL( 50,  6)
      BB=BB+   PL( 47,  1)+   PL( 48,  1)+   PL( 49,  1)+   PL( 50,  1)
      B(  2) = AA - BB
      AA=   PL(  1,  6)+   PL(  3,  6)+   PL(  7,  1)+   PL(  9,  1)
      BB=   PL(  1,  1)+   PL(  3,  1)+   PL(  7,  6)+   PL(  9,  6)
      AA=AA+   PL( 10,  1)+   PL( 12,  6)+   PL( 15,  6)+   PL( 18,  1)
      BB=BB+   PL( 10,  6)+   PL( 12,  1)+   PL( 15,  1)+   PL( 18,  6)
      BB=BB+   PL( 19,  6)+   PL( 26,  6)+   PL( 32,  1)+   PL( 33,  1)
      AA=AA+   PL( 19,  1)+   PL( 26,  1)+   PL( 32,  6)+   PL( 33,  6)
      BB=BB+   PL( 34,  1)+   PL( 35,  1)+   PL( 36,  1)+   PL( 37,  1)
      AA=AA+   PL( 34,  6)+   PL( 35,  6)+   PL( 36,  6)+   PL( 37,  6)
      AA=AA+   PL( 38,  6)
      BB=BB+   PL( 38,  1)
      B(  3) = AA - BB
      AA=   PL(  1,  1)+   PL(  2,  1)+   PL(  4,  6)+2.*PL(  5,  6)
      BB=   PL(  1,  6)+   PL(  2,  6)+   PL(  4,  1)+2.*PL(  5,  1)
      BB=BB+   PL(  8,  6)+   PL( 12,  6)+   PL( 16,  1)+2.*PL( 18,  1)
      AA=AA+   PL(  8,  1)+   PL( 12,  1)+   PL( 16,  6)+2.*PL( 18,  6)
      AA=AA+   PL( 19,  6)+   PL( 21,  1)+   PL( 22,  1)+   PL( 24,  1)
      BB=BB+   PL( 19,  1)+   PL( 21,  6)+   PL( 22,  6)+   PL( 24,  6)
      AA=AA+   PL( 27,  1)+   PL( 28,  1)+2.*PL( 32,  1)+2.*PL( 33,  1)
      BB=BB+   PL( 27,  6)+   PL( 28,  6)+2.*PL( 32,  6)+2.*PL( 33,  6)
      BB=BB+2.*PL( 34,  6)+   PL( 43,  1)+   PL( 44,  1)+   PL( 45,  1)
      AA=AA+2.*PL( 34,  1)+   PL( 43,  6)+   PL( 44,  6)+   PL( 45,  6)
      AA=AA+   PL( 46,  6)
      BB=BB+   PL( 46,  1)
      B(  4) = AA - BB
      BB=   PL(  1,  6)+   PL(  2,  1)+2.*PL(  3,  6)+   PL( 10,  1)
      AA=   PL(  1,  1)+   PL(  2,  6)+2.*PL(  3,  1)+   PL( 10,  6)
      AA=AA+   PL( 14,  6)+   PL( 15,  1)+   PL( 17,  6)+   PL( 18,  6)
      BB=BB+   PL( 14,  1)+   PL( 15,  6)+   PL( 17,  1)+   PL( 18,  1)
      BB=BB+   PL( 20,  6)+   PL( 23,  6)+   PL( 24,  1)+   PL( 25,  1)
      AA=AA+   PL( 20,  1)+   PL( 23,  1)+   PL( 24,  6)+   PL( 25,  6)
      BB=BB+   PL( 26,  1)+   PL( 27,  1)+   PL( 28,  1)+   PL( 51,  1)
      AA=AA+   PL( 26,  6)+   PL( 27,  6)+   PL( 28,  6)+   PL( 51,  6)
      AA=AA+   PL( 52,  6)+   PL( 53,  6)+   PL( 54,  6)
      BB=BB+   PL( 52,  1)+   PL( 53,  1)+   PL( 54,  1)
      B(  5) = AA - BB
      AA=   PL(  4,  1)+   PL(  6,  1)+   PL( 10,  1)+   PL( 11,  1)
      BB=   PL(  4,  6)+   PL(  6,  6)+   PL( 10,  6)+   PL( 11,  6)
      AA=AA+   PL( 12,  1)+   PL( 14,  1)+   PL( 15,  1)+   PL( 16,  1)
      BB=BB+   PL( 12,  6)+   PL( 14,  6)+   PL( 15,  6)+   PL( 16,  6)
      AA=AA+   PL( 17,  1)+   PL( 18,  1)+   PL( 20,  1)+   PL( 24,  1)
      BB=BB+   PL( 17,  6)+   PL( 18,  6)+   PL( 20,  6)+   PL( 24,  6)
      AA=AA+   PL( 51,  1)+   PL( 52,  1)+   PL( 53,  1)+   PL( 54,  1)
      BB=BB+   PL( 51,  6)+   PL( 52,  6)+   PL( 53,  6)+   PL( 54,  6)
      B(  6) = AA - BB
      AA=   PL(  4,  1)+   PL(  5,  1)+   PL(  6,  1)+   PL( 18,  1)
      BB=   PL(  4,  6)+   PL(  5,  6)+   PL(  6,  6)+   PL( 18,  6)
      AA=AA+   PL( 35,  1)+   PL( 36,  1)+   PL( 37,  1)+   PL( 38,  1)
      BB=BB+   PL( 35,  6)+   PL( 36,  6)+   PL( 37,  6)+   PL( 38,  6)
      AA=AA+   PL( 39,  1)+   PL( 40,  1)+   PL( 41,  1)+   PL( 42,  1)
      BB=BB+   PL( 39,  6)+   PL( 40,  6)+   PL( 41,  6)+   PL( 42,  6)
      AA=AA+   PL( 43,  1)+   PL( 44,  1)+   PL( 45,  1)+   PL( 46,  1)
      BB=BB+   PL( 43,  6)+   PL( 44,  6)+   PL( 45,  6)+   PL( 46,  6)
      AA=AA+   PL( 47,  1)+   PL( 48,  1)+   PL( 49,  1)+   PL( 50,  1)
      BB=BB+   PL( 47,  6)+   PL( 48,  6)+   PL( 49,  6)+   PL( 50,  6)
      AA=AA+   PL( 51,  1)+   PL( 52,  1)+   PL( 53,  1)+   PL( 54,  1)
      BB=BB+   PL( 51,  6)+   PL( 52,  6)+   PL( 53,  6)+   PL( 54,  6)
      B(  7) = AA - BB
      AA=   PL(  7,  1)+   PL( 12,  6)+   PL( 13,  6)+   PL( 14,  6)
      BB=   PL(  7,  6)+   PL( 12,  1)+   PL( 13,  1)+   PL( 14,  1)
      AA=AA+   PL( 21,  1)+   PL( 23,  1)+   PL( 26,  1)+   PL( 27,  6)
      BB=BB+   PL( 21,  6)+   PL( 23,  6)+   PL( 26,  6)+   PL( 27,  1)
      AA=AA+   PL( 47,  1)+   PL( 48,  1)+   PL( 49,  1)+   PL( 50,  1)
      BB=BB+   PL( 47,  6)+   PL( 48,  6)+   PL( 49,  6)+   PL( 50,  6)
      B(  8) = AA - BB
      BB=   PL(  4,  6)+   PL(  5,  6)+   PL( 13,  6)+   PL( 15,  1)
      AA=   PL(  4,  1)+   PL(  5,  1)+   PL( 13,  1)+   PL( 15,  6)
      AA=AA+   PL( 16,  6)+   PL( 17,  6)+   PL( 22,  1)+   PL( 27,  1)
      BB=BB+   PL( 16,  1)+   PL( 17,  1)+   PL( 22,  6)+   PL( 27,  6)
      AA=AA+   PL( 39,  1)+   PL( 40,  1)+   PL( 41,  1)+   PL( 42,  1)
      BB=BB+   PL( 39,  6)+   PL( 40,  6)+   PL( 41,  6)+   PL( 42,  6)
      B(  9) = AA - BB
      BB=   PL( 11,  1)+   PL( 15,  6)+   PL( 21,  1)+   PL( 22,  1)
      AA=   PL( 11,  6)+   PL( 15,  1)+   PL( 21,  6)+   PL( 22,  6)
      AA=AA+   PL( 23,  6)+   PL( 24,  6)+   PL( 25,  6)+   PL( 26,  6)
      BB=BB+   PL( 23,  1)+   PL( 24,  1)+   PL( 25,  1)+   PL( 26,  1)
      AA=AA+   PL( 43,  1)+   PL( 44,  1)+   PL( 45,  1)+   PL( 46,  1)
      BB=BB+   PL( 43,  6)+   PL( 44,  6)+   PL( 45,  6)+   PL( 46,  6)
      B( 10) = AA - BB
      BB=   PL(  5,  6)+   PL(  7,  1)+   PL(  8,  1)+   PL(  9,  1)
      AA=   PL(  5,  1)+   PL(  7,  6)+   PL(  8,  6)+   PL(  9,  6)
      AA=AA+   PL( 10,  6)+   PL( 19,  6)+   PL( 20,  6)+   PL( 25,  1)
      BB=BB+   PL( 10,  1)+   PL( 19,  1)+   PL( 20,  1)+   PL( 25,  6)
      AA=AA+   PL( 35,  1)+   PL( 36,  1)+   PL( 37,  1)+   PL( 38,  1)
      BB=BB+   PL( 35,  6)+   PL( 36,  6)+   PL( 37,  6)+   PL( 38,  6)
      B( 11) = AA - BB
      IF(INC)50,35,50
   35 INC=1
      DO 40 I=1, 54
      G=(RATES(2,I)+RATES(3,I)/T)/T
      PL(I,1)=G*PL(I,1)
      G=(RATES(5,I)+RATES(6,I)/T)/T
   40 PL(I,6)=G*PL(I,6)
      DO 45 I=1, 11
   45 D(I)=B(I)
      GO TO 30
   50 RETURN
      END


1119 CARDS PRINTED
```

```
$IBFTC PAE     FULIST
      SUBROUTINE PADE(A,B,C,F,N,L)
      DIMENSION A(L,L),B(L),C(L,L)
      C1=.0833333333*H*H
      C2=0.5*H
      DO 30 I=1,N
      DO 30 J=1,N
      S=0.0
   20 DO 25 K=1,N
   25 S=S+A(I,K)*A(K,J)
   30 C(I,J)=S
      DO 50 I=1,N
      B(I)=F*B(I)
      DO 40 J=1,N
   40 C(I,J)=C1*C(I,J)-C2*A(I,J)
   50 C(I,I)=C(I,I)+1.0
   60 CALL PICRTIC(A,B,C,N,L)
      RETURN
      END


  20 CARDS PRINTED
```

```
$IBFTC CRCP     FLLIST
      SUBRCUTINE PICRC(A,P,C,A,M)
      CIMENSICN A(P,P),M(1)
      CCUBLE PRECISICN CP
      P=1.C
   25 IC=C
      EPS=C.C
   3C CC 4C I=1,A
   4C EPS=EPS+ABS(A(I,I))
      EPS=1.CE-8*EPS
   55 CC 17C K=1,A
      KP=K-1
      T=-1.C
      CC 1C5 I=K,A
      IF(KP) 96,96,91
   96 CP=A(I,K)
      CC 98 J=1,KP
   98 CP=CP-A(I,J)*A(J,K)
      A(I,K)=CP
   99 IF(T-ABS(A(I,K))) 1CC,1C5,1C5
  1CC T=ABS(A(I,K))
      II=I
  1C5 CCATIAUE
      IF(II-K) 11C,135,11C
  11C IC=IC+1
      S=M(K)
      M(K)=M(II)
  12C M(II)=S
  125 CC 13C J=1,A
      S=A(K,J)
      A(K,J)=A(II,J)
  13C A(II,J)=S
  135 CT=A(K,K)
      IF(ABS(CT)-EPS) 136,136,14C
  136 WRITE(6,3CC)
  14C P=P*CT
      IF(K-A) 145,155,145
  145 KP=K+1
      CC 15C J=KP,A
      CP=A(K,J)
      IF(KP) 147,15C,147
  147 CC 148 I=1,KP
  148 CP=CP-A(K,I)*A(I,J)
  15C A(K,J)=CP/CT
  155 CP=B(K)
      IF(KP) 16C,165,16C
  16C CC 162 I=1,KP
  162 CP=CP-A(K,I)*B(I)
  165 B(K)=CP/CT
  17C CCATIAUE
      IF(MCC(IC,2)) 175,18C,175
  175 P=-P
  18C C=P
  185 II=A
      CC 19C K=2,A
      KP=II
      II=II-1
      CP=B(II)
      CC 188 I=KP,A
```

```
  188 CP=CP-A(II,I)*B(I)
  19C B(II)=CP
  20C RETURA
  30C FCRMAT(21HCREAR SINGULAR MATRIX)
      ENC

   64 CARCS PRIATEC
```

INPUT CARDS FOR GENERATING SUBROUTINES FT AND FG

(See Subsection 3.2)

```
  082566                                    11                          NK   1
7                                          11   27                      NK   2
3                                                                       NK   3
  02            0           N2         N          NO         NOX        NK   4
  E-            O6         O26         N6          N26                  NK   4.1
    1    2    3    4   79   71   72   73   74   75   76   77   78   79  NK   5
   80   81   82   83   84   85   86   87   88   89   90   91   92 5060  NK   5.1
 7000 7001 7002 7003 7004 7005 7006                                     NK   5.2
                                                                        NK   6
                                                                        NK   7
                                                                        NK   8
                                                                        NK   9
  082566                                    11                          NK   10
7                                          11                           NK   11
  1111  1    11  11 1                                                   NK   12
  02            0           N2         N          NU         NOX        NK   13
  E-            O6         O26         N6          N26                  NK   13.1
          END                                                           NK   14
```

Unclassified
Security Classification

3ND PPSO 13152

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)
General Applied Science Laboratories, Inc

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE Analysis and Application of the Pade' Approximation for the Integration of Chemical Kinetic Equations

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Technical Report - December 3, 1966 - March 17, 1967

5. AUTHOR(S) (Last name, first name, initial)
D. Magnus and H. Schechter

6. REPORT DATE
March 30, 1967

7a. TOTAL NO. OF PAGES
98

7b. NO. OF REFS
11

8a. CONTRACT OR GRANT NO.
DA-49-083 OSA-3135

b. PROJECT NO.
ARPA Order No. 396

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
TR-642

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. AVAILABILITY/LIMITATION NOTICES There shall be no dissemination or publication except within and between the Contractor and subcontractors, of information developed under this contract or contained in the reports to be furnished pursuant to this contract, without written approval of the Project Officer. Qualified requestors may obtain copies of the report from DDC.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
DOD

13. ABSTRACT A stable numerical technique is given for the integration of the fluid flow equations involving finite rate chemical kinetics. The procedure permits a significantly larger integration step than standard procedures, such as Runge-Kutta. Consequently, the amount of digital computer time to numerically integrate the equations over a specified domain is greatly reduced. To simplify the implementation of the numerical procedure for generic chemical systems, a programming system has been developed that automatically generates the necessary subroutines from a few inputs. The extension and application of the numerical method to partial differential equations is also discussed.

Unclassified
Security Classification

Unclassified
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Numerical Integration						
Chemical Kinetics						
Fluid Flow Equations						

INSTRUCTIONS

1. **ORIGINATING ACTIVITY:** Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization (*corporate author*) issuing the report.

2a. **REPORT SECURITY CLASSIFICATION:** Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. **GROUP:** Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. **REPORT TITLE:** Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. **DESCRIPTIVE NOTES:** If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. **AUTHOR(S):** Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. **REPORT DATE:** Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7a. **TOTAL NUMBER OF PAGES:** The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. **NUMBER OF REFERENCES:** Enter the total number of references cited in the report.

8a. **CONTRACT OR GRANT NUMBER:** If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. **PROJECT NUMBER:** Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. **ORIGINATOR'S REPORT NUMBER(S):** Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. **OTHER REPORT NUMBER(S):** If the report has been assigned any other report numbers (*either by the originator or by the sponsor*), also enter this number(s).

10. **AVAILABILITY/LIMITATION NOTICES:** Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through
________________________________."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through
________________________________."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through
________________________________."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. **SUPPLEMENTARY NOTES:** Use for additional explanatory notes.

12. **SPONSORING MILITARY ACTIVITY:** Enter the name of the departmental project office or laboratory sponsoring (*paying for*) the research and development. Include address.

13. **ABSTRACT:** Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. **KEY WORDS:** Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

Unclassified
Security Classification